

(old)

htmldiff from-

(new)

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349 version 21.0)

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AArch64 System Registers

[ACCDATA_EL1](#): Accelerator Data

ACTLR_EL1: Auxiliary Control Register (EL1)

ACTLR_EL2: Auxiliary Control Register (EL2)

ACTLR_EL3: Auxiliary Control Register (EL3)

[AFSR0_EL1](#): Auxiliary Fault Status Register 0 (EL1)

[AFSR0_EL2](#): Auxiliary Fault Status Register 0 (EL2)

AFSR0_EL3: Auxiliary Fault Status Register 0 (EL3)

[AFSR1_EL1](#): Auxiliary Fault Status Register 1 (EL1)

[AFSR1_EL2](#): Auxiliary Fault Status Register 1 (EL2)

AFSR1_EL3: Auxiliary Fault Status Register 1 (EL3)

[AIDR_EL1](#): Auxiliary ID Register

ALLINT: All Interrupt Mask Bit

[AMAIR2_EL1](#): Extended Auxiliary Memory Attribute Indirection Register (EL1)

[AMAIR2_EL2](#): Extended Auxiliary Memory Attribute Indirection Register (EL2)

[AMAIR2_EL3](#): Extended Auxiliary Memory Attribute Indirection Register (EL3)

[AMAIR_EL1](#): Auxiliary Memory Attribute Indirection Register (EL1)

[AMAIR_EL2](#): Auxiliary Memory Attribute Indirection Register (EL2)

AMAIR_EL3: Auxiliary Memory Attribute Indirection Register (EL3)

AMCFGR_EL0: Activity Monitors Configuration Register

AMCG1IDR_EL0: Activity Monitors Counter Group 1 Identification Register

AMCGCR_EL0: Activity Monitors Counter Group Configuration Register

[AMCNTENCLR0_EL0](#): Activity Monitors Count Enable Clear Register 0

[AMCNTENCLR1_EL0](#): Activity Monitors Count Enable Clear Register 1

[AMCNTENSET0_EL0](#): Activity Monitors Count Enable Set Register 0

[AMCNTENSET1_EL0](#): Activity Monitors Count Enable Set Register 1

AMCR_EL0: Activity Monitors Control Register

[AMEVCNTR0<n>_EL0](#): Activity Monitors Event Counter Registers 0

[AMEVCNTR1<n>_EL0](#): Activity Monitors Event Counter Registers 1

AMEVCNTVOFF0<n>_EL2: Activity Monitors Event Counter Virtual Offset Registers 0

AMEVCNTVOFF1<n>_EL2: Activity Monitors Event Counter Virtual Offset Registers 1

AMEVTYPER0<n>_EL0: Activity Monitors Event Type Registers 0

[AMEVTYPER1<n>_EL0](#): Activity Monitors Event Type Registers 1

AMUSERENR_EL0: Activity Monitors User Enable Register

[APDAKeyHi_EL1](#): Pointer Authentication Key A for Data (bits[127:64])

[APDAKeyLo_EL1](#): Pointer Authentication Key A for Data (bits[63:0])

[APDBKeyHi_EL1](#): Pointer Authentication Key B for Data (bits[127:64])

[APDBKeyLo_EL1](#): Pointer Authentication Key B for Data (bits[63:0])

[APGAKeyHi_EL1](#): Pointer Authentication Key A for Code (bits[127:64])

[APGAKeyLo_EL1](#): Pointer Authentication Key A for Code (bits[63:0])

[APIAKeyHi_EL1](#): Pointer Authentication Key A for Instruction (bits[127:64])

[APIAKeyLo_EL1](#): Pointer Authentication Key A for Instruction (bits[63:0])

[APIBKeyHi_EL1](#): Pointer Authentication Key B for Instruction (bits[127:64])

[APIBKeyLo_EL1](#): Pointer Authentication Key B for Instruction (bits[63:0])

[BRBCR_EL1](#): Branch Record Buffer Control Register (EL1)

[BRBCR_EL2](#): Branch Record Buffer Control Register (EL2)

[BRBFCR_EL1](#): Branch Record Buffer Function Control Register

[BRBIDR0_EL1](#): Branch Record Buffer ID0 Register

[BRBINF<n>_EL1](#): Branch Record Buffer Information Register <n>

[BRBINFINJ_EL1](#): Branch Record Buffer Information Injection Register

[BRBSRC<n>_EL1](#): Branch Record Buffer Source Address Register <n>

[BRBSRCINJ_EL1](#): Branch Record Buffer Source Address Injection Register

[BRBTGT<n>_EL1](#): Branch Record Buffer Target Address Register <n>

[BRBTGTINJ_EL1](#): Branch Record Buffer Target Address Injection Register

[BRBTS_EL1](#): Branch Record Buffer Timestamp Register

[CCSIDR2_EL1](#): Current Cache Size ID Register 2

[CCSIDR_EL1](#): Current Cache Size ID Register

[CLIDR_EL1](#): Cache Level ID Register

[CNTFRQ_EL0](#): Counter-timer Frequency register

[CNTHCTL_EL2](#): Counter-timer Hypervisor Control register

[CNTHPS_CTL_EL2](#): Counter-timer Secure Physical Timer Control register (EL2)

[CNTHPS_CVAL_EL2](#): Counter-timer Secure Physical Timer CompareValue register (EL2)

[CNTHPS_TVAL_EL2](#): Counter-timer Secure Physical Timer TimerValue register (EL2)

[CNTHP_CTL_EL2](#): Counter-timer Hypervisor Physical Timer Control register

[CNTHP_CVAL_EL2](#): Counter-timer Physical Timer CompareValue register (EL2)

[CNTHP_TVAL_EL2](#): Counter-timer Physical Timer TimerValue register (EL2)

[CNTHVS_CTL_EL2](#): Counter-timer Secure Virtual Timer Control register (EL2)

[CNTHVS_CVAL_EL2](#): Counter-timer Secure Virtual Timer CompareValue register (EL2)

[CNTHVS_TVAL_EL2](#): Counter-timer Secure Virtual Timer TimerValue register (EL2)

[CNTHV_CTL_EL2](#): Counter-timer Virtual Timer Control register (EL2)

CNTHV_CVAL_EL2: Counter-timer Virtual Timer CompareValue register (EL2)

CNTHV_TVAL_EL2: Counter-timer Virtual Timer TimerValue Register (EL2)

CNTKCTL_EL1: Counter-timer Kernel Control register

CNTPCTSS_EL0: Counter-timer Self-Synchronized Physical Count register

CNTPCT_EL0: Counter-timer Physical Count register

CNTPOFF_EL2: Counter-timer Physical Offset register

[CNTPS_CTL_EL1](#): Counter-timer Physical Secure Timer Control register

[CNTPS_CVAL_EL1](#): Counter-timer Physical Secure Timer CompareValue register

[CNTPS_TVAL_EL1](#): Counter-timer Physical Secure Timer TimerValue register

CNTP_CTL_EL0: Counter-timer Physical Timer Control register

CNTP_CVAL_EL0: Counter-timer Physical Timer CompareValue register

CNTP_TVAL_EL0: Counter-timer Physical Timer TimerValue register

CNTVCTSS_EL0: Counter-timer Self-Synchronized Virtual Count register

CNTVCT_EL0: Counter-timer Virtual Count register

CNTVOFF_EL2: Counter-timer Virtual Offset register

CNTV_CTL_EL0: Counter-timer Virtual Timer Control register

CNTV_CVAL_EL0: Counter-timer Virtual Timer CompareValue register

CNTV_TVAL_EL0: Counter-timer Virtual Timer TimerValue register

[CONTEXTIDR_EL1](#): Context ID Register (EL1)

[CONTEXTIDR_EL2](#): Context ID Register (EL2)

[CPACR_EL1](#): Architectural Feature Access Control Register

[CPTR_EL2](#): Architectural Feature Trap Register (EL2)

[CPTR_EL3](#): Architectural Feature Trap Register (EL3)

[CSSELR_EL1](#): Cache Size Selection Register

[CTR_EL0](#): Cache Type Register

CurrentEL: Current Exception Level

DACR32_EL2: Domain Access Control Register

DAIF: Interrupt Mask Bits

[DBGAUTHSTATUS_EL1](#): Debug Authentication Status register

[DBGBCR<n>_EL1](#): Debug Breakpoint Control Registers

[DBGBVR<n>_EL1](#): Debug Breakpoint Value Registers

[DBGCLAIMCLR_EL1](#): Debug CLAIM Tag Clear register

[DBGCLAIMSET_EL1](#): Debug CLAIM Tag Set register

DBGDTRRX_EL0: Debug Data Transfer Register, Receive

DBGDTRTX_EL0: Debug Data Transfer Register, Transmit

DBGDTR_EL0: Debug Data Transfer Register, half-duplex

[DBGPRCR_EL1](#): Debug Power Control Register

DBGVCR32_EL2: Debug Vector Catch Register

[DBGWCR<n>_EL1](#): Debug Watchpoint Control Registers

[DBGWVR<n>_EL1](#): Debug Watchpoint Value Registers

[DCZID_EL0](#): Data Cache Zero ID register

DISR_EL1: Deferred Interrupt Status Register

DIT: Data Independent Timing

DLR_EL0: Debug Link Register

[DSPSR_EL0](#): Debug Saved Program Status Register

ELR_EL1: Exception Link Register (EL1)

ELR_EL2: Exception Link Register (EL2)

ELR_EL3: Exception Link Register (EL3)

[ERRIDR_EL1](#): Error Record ID Register

[ERRSELR_EL1](#): Error Record Select Register

[ERXADDR_EL1](#): Selected Error Record Address Register

[ERXCTLR_EL1](#): Selected Error Record Control Register

[ERXFR_EL1](#): Selected Error Record Feature Register

[ERXMISC0_EL1](#): Selected Error Record Miscellaneous Register 0

[ERXMISC1_EL1](#): Selected Error Record Miscellaneous Register 1

[ERXMISC2_EL1](#): Selected Error Record Miscellaneous Register 2

[ERXMISC3_EL1](#): Selected Error Record Miscellaneous Register 3

[ERXPFGCDN_EL1](#): Selected Pseudo-fault Generation Countdown register

[ERXPFGCTL_EL1](#): Selected Pseudo-fault Generation Control register

[ERXPFGF_EL1](#): Selected Pseudo-fault Generation Feature register

[ERXSTATUS_EL1](#): Selected Error Record Primary Status Register

[ESR_EL1](#): Exception Syndrome Register (EL1)

[ESR_EL2](#): Exception Syndrome Register (EL2)

[ESR_EL3](#): Exception Syndrome Register (EL3)

[FAR_EL1](#): Fault Address Register (EL1)

[FAR_EL2](#): Fault Address Register (EL2)

[FAR_EL3](#): Fault Address Register (EL3)

[FPCR](#): Floating-point Control Register

FPEXC32_EL2: Floating-Point Exception Control register

FPSR: Floating-point Status Register

GCR_EL1: Tag Control Register.

GMID_EL1: Multiple tag transfer ID register

[GPCCR_EL3](#): Granule Protection Check Control Register (EL3)

GPTBR_EL3: Granule Protection Table Base Register

HACR_EL2: Hypervisor Auxiliary Control Register

HAFGRTR_EL2: Hypervisor Activity Monitors Fine-Grained Read Trap Register

[HCRX_EL2](#): Extended Hypervisor Configuration Register

[HCR_EL2](#): Hypervisor Configuration Register

[HDFGRTR2_EL2](#): Hypervisor Debug Fine-Grained Read Trap Register 2

[HDFGRTR_EL2](#): Hypervisor Debug Fine-Grained Read Trap Register

[HDFGWTR2_EL2](#): Hypervisor Debug Fine-Grained Write Trap Register 2

[HDFGWTR_EL2](#): Hypervisor Debug Fine-Grained Write Trap Register

[HFGITR2_EL2](#): Hypervisor Fine-Grained Instruction Trap Register 2

[HFGITR_EL2](#): Hypervisor Fine-Grained Instruction Trap Register

[HFGRTR2_EL2](#): Hypervisor Fine-Grained Read Trap Register 2

[HFGRTR_EL2](#): Hypervisor Fine-Grained Read Trap Register

[HFGWTR2_EL2](#): Hypervisor Fine-Grained Write Trap Register 2

[HFGWTR_EL2](#): Hypervisor Fine-Grained Write Trap Register

[HPFAR_EL2](#): Hypervisor IPA Fault Address Register

HSTR_EL2: Hypervisor System Trap Register

ICC_AP0R<n>_EL1: Interrupt Controller Active Priorities Group 0 Registers

[ICC_AP1R<n>_EL1](#): Interrupt Controller Active Priorities Group 1 Registers

ICC_ASGI1R_EL1: Interrupt Controller Alias Software Generated Interrupt Group 1 Register

ICC_BPR0_EL1: Interrupt Controller Binary Point Register 0

[ICC_BPR1_EL1](#): Interrupt Controller Binary Point Register 1

[ICC_CTLR_EL1](#): Interrupt Controller Control Register (EL1)

[ICC_CTLR_EL3](#): Interrupt Controller Control Register (EL3)

ICC_DIR_EL1: Interrupt Controller Deactivate Interrupt Register

ICC_EOIR0_EL1: Interrupt Controller End Of Interrupt Register 0

ICC_EOIR1_EL1: Interrupt Controller End Of Interrupt Register 1

ICC_HPPIR0_EL1: Interrupt Controller Highest Priority Pending Interrupt Register 0

ICC_HPPIR1_EL1: Interrupt Controller Highest Priority Pending Interrupt Register 1

ICC_IAR0_EL1: Interrupt Controller Interrupt Acknowledge Register 0

ICC_IAR1_EL1: Interrupt Controller Interrupt Acknowledge Register 1

[ICC_IGRPEN0_EL1](#): Interrupt Controller Interrupt Group 0 Enable register

[ICC_IGRPEN1_EL1](#): Interrupt Controller Interrupt Group 1 Enable register

ICC_IGRPEN1_EL3: Interrupt Controller Interrupt Group 1 Enable register (EL3)

ICC_NMIAR1_EL1: Interrupt Controller Non-maskable Interrupt Acknowledge Register 1

ICC_PMR_EL1: Interrupt Controller Interrupt Priority Mask Register

ICC_RPR_EL1: Interrupt Controller Running Priority Register

ICC_SGI0R_EL1: Interrupt Controller Software Generated Interrupt Group 0 Register

ICC_SGI1R_EL1: Interrupt Controller Software Generated Interrupt Group 1 Register

[ICC_SRE_EL1](#): Interrupt Controller System Register Enable register (EL1)

ICC_SRE_EL2: Interrupt Controller System Register Enable register (EL2)

ICC_SRE_EL3: Interrupt Controller System Register Enable register (EL3)

ICH_AP0R<n>_EL2: Interrupt Controller Hyp Active Priorities Group 0 Registers

ICH_AP1R<n>_EL2: Interrupt Controller Hyp Active Priorities Group 1 Registers

ICH_EISR_EL2: Interrupt Controller End of Interrupt Status Register

ICH_ELRSR_EL2: Interrupt Controller Empty List Register Status Register

ICH_HCR_EL2: Interrupt Controller Hyp Control Register

ICH_LR<n>_EL2: Interrupt Controller List Registers

ICH_MISR_EL2: Interrupt Controller Maintenance Interrupt State Register

ICH_VMCR_EL2: Interrupt Controller Virtual Machine Control Register

ICH_VTR_EL2: Interrupt Controller VGIC Type Register

ICV_AP0R<n>_EL1: Interrupt Controller Virtual Active Priorities Group 0 Registers

ICV_AP1R<n>_EL1: Interrupt Controller Virtual Active Priorities Group 1 Registers

ICV_BPR0_EL1: Interrupt Controller Virtual Binary Point Register 0

ICV_BPR1_EL1: Interrupt Controller Virtual Binary Point Register 1

[ICV_CTLR_EL1](#): Interrupt Controller Virtual Control Register

ICV_DIR_EL1: Interrupt Controller Deactivate Virtual Interrupt Register

ICV_EOIR0_EL1: Interrupt Controller Virtual End Of Interrupt Register 0

ICV_EOIR1_EL1: Interrupt Controller Virtual End Of Interrupt Register 1

ICV_HPPIR0_EL1: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

ICV_HPPIR1_EL1: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

ICV_IAR0_EL1: Interrupt Controller Virtual Interrupt Acknowledge Register 0

ICV_IAR1_EL1: Interrupt Controller Virtual Interrupt Acknowledge Register 1

[ICV_IGRPEN0_EL1](#): Interrupt Controller Virtual Interrupt Group 0 Enable register

[ICV_IGRPEN1_EL1](#): Interrupt Controller Virtual Interrupt Group 1 Enable register

ICV_NMIAR1_EL1: Interrupt Controller Virtual Non-maskable Interrupt Acknowledge Register 1

ICV_PMR_EL1: Interrupt Controller Virtual Interrupt Priority Mask Register

ICV_RPR_EL1: Interrupt Controller Virtual Running Priority Register

ID_AA64AFR0_EL1: AArch64 Auxiliary Feature Register 0

ID_AA64AFR1_EL1: AArch64 Auxiliary Feature Register 1

[ID_AA64DFR0_EL1](#): AArch64 Debug Feature Register 0

[ID_AA64DFR1_EL1](#): AArch64 Debug Feature Register 1

[ID_AA64ISAR0_EL1](#): AArch64 Instruction Set Attribute Register 0

[ID_AA64ISAR1_EL1](#): AArch64 Instruction Set Attribute Register 1

[ID_AA64ISAR2_EL1](#): AArch64 Instruction Set Attribute Register 2

[ID_AA64MMFR0_EL1](#): AArch64 Memory Model Feature Register 0

[ID_AA64MMFR1_EL1](#): AArch64 Memory Model Feature Register 1

[ID_AA64MMFR2_EL1](#): AArch64 Memory Model Feature Register 2

[ID_AA64MMFR3_EL1](#): AArch64 Memory Model Feature Register 3

[ID_AA64MMFR4_EL1](#): AArch64 Memory Model Feature Register 4

[ID_AA64PFR0_EL1](#): AArch64 Processor Feature Register 0

[ID_AA64PFR1_EL1](#): AArch64 Processor Feature Register 1

[ID_AA64PFR2_EL1](#): AArch64 Processor Feature Register 2

[ID_AA64SMFR0_EL1](#): SME Feature ID register 0

[ID_AA64ZFR0_EL1](#): SVE Feature ID register 0

ID_AFR0_EL1: AArch32 Auxiliary Feature Register 0

[ID_DFR0_EL1](#): AArch32 Debug Feature Register 0

ID_DFR1_EL1: Debug Feature Register 1

ID_ISAR0_EL1: AArch32 Instruction Set Attribute Register 0

ID_ISAR1_EL1: AArch32 Instruction Set Attribute Register 1

ID_ISAR2_EL1: AArch32 Instruction Set Attribute Register 2

ID_ISAR3_EL1: AArch32 Instruction Set Attribute Register 3

ID_ISAR4_EL1: AArch32 Instruction Set Attribute Register 4

ID_ISAR5_EL1: AArch32 Instruction Set Attribute Register 5

[ID_ISAR6_EL1](#): AArch32 Instruction Set Attribute Register 6

ID_MMFR0_EL1: AArch32 Memory Model Feature Register 0

ID_MMFR1_EL1: AArch32 Memory Model Feature Register 1

ID_MMFR2_EL1: AArch32 Memory Model Feature Register 2

ID_MMFR3_EL1: AArch32 Memory Model Feature Register 3

ID_MMFR4_EL1: AArch32 Memory Model Feature Register 4

ID_MMFR5_EL1: AArch32 Memory Model Feature Register 5

[ID_PFR0_EL1](#): AArch32 Processor Feature Register 0

ID_PFR1_EL1: AArch32 Processor Feature Register 1

ID_PFR2_EL1: AArch32 Processor Feature Register 2

IFSR32_EL2: Instruction Fault Status Register (EL2)

[ISR_EL1](#): Interrupt Status Register

[LORC_EL1](#): LORegion Control (EL1)

[LOREA_EL1](#): LORegion End Address (EL1)

[LORID_EL1](#): LORegionID (EL1)

[LORN_EL1](#): LORegion Number (EL1)

[LORSA_EL1](#): LORegion Start Address (EL1)

[MAIR2_EL1](#): Extended Memory Attribute Indirection Register (EL1)

[MAIR2_EL2](#): Extended Memory Attribute Indirection Register (EL2)

[MAIR2_EL3](#): Extended Memory Attribute Indirection Register (EL3)

[MAIR_EL1](#): Memory Attribute Indirection Register (EL1)

[MAIR_EL2](#): Memory Attribute Indirection Register (EL2)

[MAIR_EL3](#): Memory Attribute Indirection Register (EL3)

MDCCINT_EL1: Monitor DCC Interrupt Enable Register

MDCCSR_EL0: Monitor DCC Status Register

[MDCR_EL2](#): Monitor Debug Configuration Register (EL2)

[MDCR_EL3](#): Monitor Debug Configuration Register (EL3)

[MDRAR_EL1](#): Monitor Debug ROM Address Register

[MDSCR_EL1](#): Monitor Debug System Control Register

[MDSELR_EL1](#)[MFAR_EL3](#): Breakpoint PA and Fault Watchpoint Selection Address Register

[MECIDR_EL2](#): MEC Identification Register

[MECID_A0_EL2](#): Alternate MECID for EL2 and EL2&0 translation regimes

[MECID_A1_EL2](#): Alternate MECID for EL2&0 translation regimes.

[MECID_P0_EL2](#): Primary MECID for EL2 and EL2&0 translation regimes

[MECID_P1_EL2](#): Primary MECID for EL2&0 translation regimes

[MECID_RL_A_EL3](#): Realm PA space Alternate MECID for EL3 stage 1 translation regime

[MFAR_EL3](#): Physical Fault Address Register (EL3)

[MIDR_EL1](#): Main ID Register

MPAM0_EL1: MPAM0 Register (EL1)

MPAM1_EL1: MPAM1 Register (EL1)

MPAM2_EL2: MPAM2 Register (EL2)

MPAM3_EL3: MPAM3 Register (EL3)

MPAMHCR_EL2: MPAM Hypervisor Control Register (EL2)

MPAMIDR_EL1: MPAM ID Register (EL1)

MPAMSM_EL1: MPAM Streaming Mode Register

MPAMVPM0_EL2: MPAM Virtual PARTID Mapping Register 0

MPAMVPM1_EL2: MPAM Virtual PARTID Mapping Register 1

MPAMVPM2_EL2: MPAM Virtual PARTID Mapping Register 2

MPAMVPM3_EL2: MPAM Virtual PARTID Mapping Register 3

MPAMVPM4_EL2: MPAM Virtual PARTID Mapping Register 4

MPAMVPM5_EL2: MPAM Virtual PARTID Mapping Register 5

MPAMVPM6_EL2: MPAM Virtual PARTID Mapping Register 6

MPAMVPM7_EL2: MPAM Virtual PARTID Mapping Register 7

MPAMVPMV_EL2: MPAM Virtual Partition Mapping Valid Register

[MPIDR_EL1](#): Multiprocessor Affinity Register

MVFR0_EL1: AArch32 Media and VFP Feature Register 0

MVFR1_EL1: AArch32 Media and VFP Feature Register 1

MVFR2_EL1: AArch32 Media and VFP Feature Register 2

NZCV: Condition Flags

[OSDLR_EL1](#): OS Double Lock Register

OSDTRRX_EL1: OS Lock Data Transfer Register, Receive

OSDTRTX_EL1: OS Lock Data Transfer Register, Transmit

[OSECCR_EL1](#): OS Lock Exception Catch Control Register

[OSLAR_EL1](#): OS Lock Access Register

[OSLSR_EL1](#): OS Lock Status Register

PAN: Privileged Access Never

[PAR_EL1](#): Physical Address Register

[PFAR_EL1](#): Physical Fault Address Register (EL1)

[PFAR_EL2](#): Physical Fault Address Register (EL2)

[PIRE0_EL1](#): Permission Indirection Register 0 (EL1)

[PIRE0_EL2](#): Permission Indirection Register 0 (EL2)

[PIR_EL1](#): Permission Indirection Register 1 (EL1)

[PIR_EL2](#): Permission Indirection Register 2 (EL2)

[PIR_EL3](#): Permission Indirection Register 3 (EL3)

[PM](#): PMU Exception Mask

[PMBIDR_EL1](#): Profiling Buffer ID Register

[PMBLIMTR_EL1](#): Profiling Buffer Limit Address Register

[PMBPTR_EL1](#): Profiling Buffer Write Pointer Register

[PMBSR_EL1](#): Profiling Buffer Status/syndrome Register

[PMCCFILTR_EL0](#): Performance Monitors Cycle Count Filter Register

[PMCCNTR_EL0](#): Performance Monitors Cycle Count Register

[PMCCNTSVR_EL1](#): Performance Monitors Cycle Count Saved Value Register

[PMCEID0_EL0](#): Performance Monitors Common Event Identification register 0

[PMCEID1_EL0](#): Performance Monitors Common Event Identification register 1

[PMCNTENCLR_EL0](#): Performance Monitors Count Enable Clear register

[PMCNTENSET_EL0](#): Performance Monitors Count Enable Set register

[PMCR_EL0](#): Performance Monitors Control Register

[PMECR_EL1](#): Performance Monitors Exception Control Register

[PMEVCNTR<n>_EL0](#): Performance Monitors Event Count Registers

[PMEVCNTSVR<n>_EL1](#): Performance Monitors Event Count Saved Value Register <n>

[PMEVTYPER<n>_EL0](#): Performance Monitors Event Type Registers

[PMIAR_EL1](#): Performance Monitors Instruction Address Register

[PMICFILTR_EL0](#): Performance Monitors Instruction Counter Filter Register

[PMICNTR_EL0](#): Performance Monitors Instruction Counter Register

[PMICNTSVR_EL1](#): Performance Monitors Instruction Count Saved Value Register

[PMINTENCLR_EL1](#): Performance Monitors Interrupt Enable Clear register

[PMINTENSET_EL1](#): Performance Monitors Interrupt Enable Set register

[PMMIR_EL1](#): Performance Monitors Machine Identification Register

[PMOVSCLR_EL0](#): Performance Monitors Overflow Flag Status Clear Register

[PMOVSSET_EL0](#): Performance Monitors Overflow Flag Status Set register

[PMSCR_EL1](#): Statistical Profiling Control Register (EL1)

[PMSCR_EL2](#): Statistical Profiling Control Register (EL2)

[PMSDSFR_EL1](#): Sampling Data Source Filter Register

[PMSELR_EL0](#): Performance Monitors Event Counter Selection Register

[PMSEVFR_EL1](#): Sampling Event Filter Register

[PMSFCR_EL1](#): Sampling Filter Control Register

[PMSICR_EL1](#): Sampling Interval Counter Register

[PMSIDR_EL1](#): Sampling Profiling ID Register

[PMSIRR_EL1](#): Sampling Interval Reload Register

[PMSLATFR_EL1](#): Sampling Latency Filter Register

[PMSNEVFR_EL1](#): Sampling Inverted Event Filter Register

[PMSSCR_EL1](#): Performance Monitors Snapshot Status and Capture Register

[PMSWINC_EL0](#): Performance Monitors Software Increment register

[PMUACR_EL1](#): Performance Monitors User Access Control Register

[PMUSERENR_EL0](#): Performance Monitors User Enable Register

[PMXEVCNTR_EL0](#): Performance Monitors Selected Event Count Register

[PMXEVTYPER_EL0](#): Performance Monitors Selected Event Type Register

[PMZR_EL0](#): Performance Monitors Zero with Mask

[POR_EL0](#): Permission Overlay Register 0 (EL0)

[POR_EL1](#): Permission Overlay Register 1 (EL1)

[POR_EL2](#): Permission Overlay Register 2 (EL2)

POR_EL3: Permission Overlay Register 3 (EL3)

RCWMASK_EL1: Read Check Write Instruction Mask (EL1)

RCWSMASK_EL1: Software Read Check Write Instruction Mask (EL1)

REVIDR_EL1: Revision ID Register

RGSR_EL1: Random Allocation Tag Seed Register.

RMR_EL1: Reset Management Register (EL1)

RMR_EL2: Reset Management Register (EL2)

RMR_EL3: Reset Management Register (EL3)

RNDR: Random Number

RNDRRS: Reseeded Random Number

RVBAR_EL1: Reset Vector Base Address Register (if EL2 and EL3 not implemented)

RVBAR_EL2: Reset Vector Base Address Register (if EL3 not implemented)

RVBAR_EL3: Reset Vector Base Address Register (if EL3 implemented)

S2PIR_EL2: Stage 2 Permission Indirection Register (EL2)

S2POR_EL1: Stage 2 Permission Overlay Register (EL1)

S3_<op1>_<Cn>_<Cm>_<op2>: IMPLEMENTATION DEFINED registers

SCR_EL3: Secure Configuration Register

SCTLR2_EL1: System Control Register (EL1)

SCTLR2_EL2: System Control Register (EL2)

SCTLR2_EL3: System Control Register (EL3)

SCTLR_EL1: System Control Register (EL1)

SCTLR_EL2: System Control Register (EL2)

SCTLR_EL3: System Control Register (EL3)

SCXTNUM_EL0: EL0 Read/Write Software Context Number

SCXTNUM_EL1: EL1 Read/Write Software Context Number

SCXTNUM_EL2: EL2 Read/Write Software Context Number

SCXTNUM_EL3: EL3 Read/Write Software Context Number

SDER32_EL2: AArch32 Secure Debug Enable Register

SDER32_EL3: AArch32 Secure Debug Enable Register

SMCR_EL1: SME Control Register (EL1)

SMCR_EL2: SME Control Register (EL2)

SMCR_EL3: SME Control Register (EL3)

SMIDR_EL1: Streaming Mode Identification Register

SMPRIMAP_EL2: Streaming Mode Priority Mapping Register

SMPRI_EL1: Streaming Mode Priority Register

SPMACCESSR_EL1: System Performance Monitors Access Register (EL1)

[SPMACCESSR_EL2](#): System Performance Monitors Access Register (EL2)

[SPMACCESSR_EL3](#): System Performance Monitors Access Register (EL3)

[SPMCFGR_EL1](#): System Performance Monitors Configuration Register

[SPMCGCR<n>_EL1](#): Counter Group Configuration Register <n>

[SPMCNTENCLR_EL0](#): System Performance Monitors Count Enable Clear Register

[SPMCNTENSET_EL0](#): System Performance Monitors Count Enable Set Register

[SPMCR_EL0](#): System Performance Monitor Control Register

[SPMDEVAFF_EL1](#): System Performance Monitors Device Affinity Register

[SPMDEVARCH_EL1](#): System Performance Monitors Device Architecture Register

[SPMEVCNTR<n>_EL0](#): System Performance Monitors Event Count Register

[SPMEVFILT2R<n>_EL0](#): Additional System Performance Monitors Event Filter Control Register 2

[SPMEVFILTR<n>_EL0](#): Additional System Performance Monitors Event Filter Control Register

[SPMEVTYPER<n>_EL0](#): System Performance Monitors Event Type Register

[SPMIIDR_EL1](#): Implementation Identification Register

[SPMINTENCLR_EL1](#): System Performance Monitors Interrupt Enable Clear Register

[SPMINTENSET_EL1](#): System Performance Monitors Interrupt Enable Set Register

[SPMOVSCLR_EL0](#): System Performance Monitors Overflow Flag Status Clear Register

[SPMOVSSSET_EL0](#): System Performance Monitors Overflow Flag Status Set Register

[SPMROOTCR_EL3](#): System Performance Monitors Root and Realm Control Register

[SPMSCR_EL1](#): System Performance Monitors Secure Control Register

[SPMSELR_EL0](#): System Performance Monitors Select Register

SPSel: Stack Pointer Select

SPSR_abt: Saved Program Status Register (Abort mode)

[SPSR_EL1](#): Saved Program Status Register (EL1)

[SPSR_EL2](#): Saved Program Status Register (EL2)

[SPSR_EL3](#): Saved Program Status Register (EL3)

SPSR_fiq: Saved Program Status Register (FIQ mode)

SPSR_irq: Saved Program Status Register (IRQ mode)

SPSR_und: Saved Program Status Register (Undefined mode)

SP_EL0: Stack Pointer (EL0)

SP_EL1: Stack Pointer (EL1)

SP_EL2: Stack Pointer (EL2)

SP_EL3: Stack Pointer (EL3)

SSBS: Speculative Store Bypass Safe

[SVCR](#): Streaming Vector Control Register

[TCO](#): Tag Check Override

[TCR2_EL1](#): Extended Translation Control Register (EL1)

[TCR2_EL2](#): Extended Translation Control Register (EL2)

[TCR_EL1](#): Translation Control Register (EL1)

[TCR_EL2](#): Translation Control Register (EL2)

[TCR_EL3](#): Translation Control Register (EL3)

TFSRE0_EL1: Tag Fault Status Register (EL0).

TFSR_EL1: Tag Fault Status Register (EL1)

TFSR_EL2: Tag Fault Status Register (EL2)

TFSR_EL3: Tag Fault Status Register (EL3)

[TPIDR2_EL0](#): EL0 Read/Write Software Thread ID Register 2

[TPIDRRO_EL0](#): EL0 Read-Only Software Thread ID Register

[TPIDR_EL0](#): EL0 Read/Write Software Thread ID Register

[TPIDR_EL1](#): EL1 Software Thread ID Register

TPIDR_EL2: EL2 Software Thread ID Register

TPIDR_EL3: EL3 Software Thread ID Register

[TRBBASER_EL1](#): Trace Buffer Base Address Register

[TRBIDR_EL1](#): Trace Buffer ID Register

[TRBLIMITR_EL1](#): Trace Buffer Limit Address Register

[TRBMAR_EL1](#): Trace Buffer Memory Attribute Register

[TRBPTR_EL1](#): Trace Buffer Write Pointer Register

[TRBSR_EL1](#): Trace Buffer Status/syndrome Register

[TRBTRG_EL1](#): Trace Buffer Trigger Counter Register

[TRCACATR<n>](#): Address Comparator Access Type Register <n>

[TRCACVR<n>](#): Address Comparator Value Register <n>

[TRCAUTHSTATUS](#): Authentication Status Register

[TRCAUXCTLR](#): Auxiliary Control Register

[TRCBBCTLR](#): Branch Broadcast Control Register

[TRCCCCTLR](#): Cycle Count Control Register

[TRCCIDCCTLR0](#): Context Identifier Comparator Control Register 0

[TRCCIDCCTLR1](#): Context Identifier Comparator Control Register 1

[TRCCIDCVR<n>](#): Context Identifier Comparator Value Registers <n>

[TRCCLAIMCLR](#): Claim Tag Clear Register

[TRCCLAIMSET](#): Claim Tag Set Register

[TRCCNTCTLR<n>](#): Counter Control Register <n>

[TRCCNTRLDVR<n>](#): Counter Reload Value Register <n>

[TRCCNTVR<n>](#): Counter Value Register <n>

[TRCCONFIGR](#): Trace Configuration Register

[TRCDEVARCH](#): Device Architecture Register

[TRCDEVID](#): Device Configuration Register

[TRCEVENTCTL0R](#): Event Control 0 Register

[TRCEVENTCTL1R](#): Event Control 1 Register

[TRCEXTINSELR<n>](#): External Input Select Register <n>

[TRCIDR0](#): ID Register 0

[TRCIDR1](#): ID Register 1

[TRCIDR10](#): ID Register 10

[TRCIDR11](#): ID Register 11

[TRCIDR12](#): ID Register 12

[TRCIDR13](#): ID Register 13

[TRCIDR2](#): ID Register 2

[TRCIDR3](#): ID Register 3

[TRCIDR4](#): ID Register 4

[TRCIDR5](#): ID Register 5

[TRCIDR6](#): ID Register 6

[TRCIDR7](#): ID Register 7

[TRCIDR8](#): ID Register 8

[TRCIDR9](#): ID Register 9

[TRCIMSPEC0](#): IMP DEF Register 0

[TRCIMSPEC<n>](#): IMP DEF Register <n>

[TRCITECR_EL1](#): Instrumentation Trace Control Register (EL1)

[TRCITECR_EL2](#): Instrumentation Trace Control Register (EL2)

[TRCITEEDCR](#): Instrumentation Trace Extension External Debug Control Register

[TRCOSLSR](#): Trace OS Lock Status Register

[TRCPRGCTLR](#): Programming Control Register

[TRCQCTLR](#): Q Element Control Register

[TRCRSCTLR<n>](#): Resource Selection Control Register <n>

[TRCRSR](#): Resources Status Register

[TRCSEQEVR<n>](#): Sequencer State Transition Control Register <n>

[TRCSEQRSTEV](#): Sequencer Reset Control Register

[TRCSEQSTR](#): Sequencer State Register

[TRCSSCCR<n>](#): Single-shot Comparator Control Register <n>

[TRCSSCSR<n>](#): Single-shot Comparator Control Status Register <n>

[TRCSSPCICR<n>](#): Single-shot Processing Element Comparator Input Control Register <n>

[TRCSTALLCTLR](#): Stall Control Register

[TRCSTATR](#): Trace Status Register

[TRCSYNCPR](#): Synchronization Period Register

[TRCTRACEIDR](#): Trace ID Register

[TRCTSCTLR](#): Timestamp Control Register

[TRCVICTLR](#): ViewInst Main Control Register

[TRCVIIECTLR](#): ViewInst Include/Exclude Control Register

[TRCVIPCSSCTLR](#): ViewInst Start/Stop PE Comparator Control Register

[TRCVISSCTLR](#): ViewInst Start/Stop Control Register

[TRCVMIDCCTLR0](#): Virtual Context Identifier Comparator Control Register 0

[TRCVMIDCCTLR1](#): Virtual Context Identifier Comparator Control Register 1

[TRCVMIDCVR<n>](#): Virtual Context Identifier Comparator Value Register <n>

[TRFCR_EL1](#): Trace Filter Control Register (EL1)

[TRFCR_EL2](#): Trace Filter Control Register (EL2)

[TTBR0_EL1](#): Translation Table Base Register 0 (EL1)

[TTBR0_EL2](#): Translation Table Base Register 0 (EL2)

[TTBR0_EL3](#): Translation Table Base Register 0 (EL3)

[TTBR1_EL1](#): Translation Table Base Register 1 (EL1)

[TTBR1_EL2](#): Translation Table Base Register 1 (EL2)

UAO: User Access Override

[VBAR_EL1](#): Vector Base Address Register (EL1)

[VBAR_EL2](#): Vector Base Address Register (EL2)

[VBAR_EL3](#): Vector Base Address Register (EL3)

VDISR_EL2: Virtual Deferred Interrupt Status Register

[VMECID_A_EL2](#): Alternate MECID for EL1&0 stage 2 translation regime

[VMECID_P_EL2](#): Primary MECID for EL1&0 stage 2 translation regime

[VMPIDR_EL2](#): Virtualization Multiprocessor ID Register

[VNCR_EL2](#): Virtual Nested Control Register

[VPIDR_EL2](#): Virtualization Processor ID Register

VSESR_EL2: Virtual SError Exception Syndrome Register

[VSTCR_EL2](#): Virtualization Secure Translation Control Register

[VSTTBR_EL2](#): Virtualization Secure Translation Table Base Register

[VTCCR_EL2](#): Virtualization Translation Control Register

[VTTBR_EL2](#): Virtualization Translation Table Base Register

[ZCR_EL1](#): SVE Control Register (EL1)

[ZCR_EL2](#): SVE Control Register (EL2)

[ZCR_EL3](#): SVE Control Register (EL3)

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AArch64 System Instructions

AT S12E0R: Address Translate Stages 1 and 2 EL0 Read

AT S12E0W: Address Translate Stages 1 and 2 EL0 Write

AT S12E1R: Address Translate Stages 1 and 2 EL1 Read

AT S12E1W: Address Translate Stages 1 and 2 EL1 Write

[AT S1E0R](#): Address Translate Stage 1 EL0 Read

[AT S1E0W](#): Address Translate Stage 1 EL0 Write

[AT S1E1R](#): Address Translate Stage 1 EL1 Read

[AT S1E1RP](#): Address Translate Stage 1 EL1 Read PAN

[AT S1E1W](#): Address Translate Stage 1 EL1 Write

[AT S1E1WP](#): Address Translate Stage 1 EL1 Write PAN

AT S1E2R: Address Translate Stage 1 EL2 Read

AT S1E2W: Address Translate Stage 1 EL2 Write

AT S1E3R: Address Translate Stage 1 EL3 Read

AT S1E3W: Address Translate Stage 1 EL3 Write

[BRB IALL](#): Invalidate the Branch Record Buffer

[BRB INJ](#): Branch Record Injection into the Branch Record Buffer

[CFP RCTX](#): Control Flow Prediction Restriction by Context

[COSP RCTX](#): Clear Other Speculative Restriction by Context

[CPP RCTX](#): Cache Prefetch Prediction Restriction by Context

[DC CGDSW](#): Clean of Data and Allocation Tags by Set/Way

[DC CGDVAC](#): Clean of Data and Allocation Tags by VA to PoC

[DC CGDVADP](#): Clean of Data and Allocation Tags by VA to PoDP

[DC CGDVAP](#): Clean of Data and Allocation Tags by VA to PoP

[DC CGSW](#): Clean of Allocation Tags by Set/Way

[DC CGVAC](#): Clean of Allocation Tags by VA to PoC

[DC CGVADP](#): Clean of Allocation Tags by VA to PoDP

[DC CGVAP](#): Clean of Allocation Tags by VA to PoP

[DC CIGDPAE](#): Clean and invalidate of data and allocation tags by PA to PoE

[DC CIGDPAPA](#): Clean and Invalidate of Data and Allocation Tags by PA to PoPA

[DC CIGDSW](#): Clean and Invalidate of Data and Allocation Tags by Set/Way

[DC CIGDVAC](#): Clean and Invalidate of Data and Allocation Tags by VA to PoC

[DC CIGSW](#): Clean and Invalidate of Allocation Tags by Set/Way

[DC CIGVAC](#): Clean and Invalidate of Allocation Tags by VA to PoC

DC CIPAE: Data or unified Cache line Clean and Invalidate by PA to PoE

DC CIPAPA: Data or unified Cache line Clean and Invalidate by PA to PoPA

DC CISW: Data or unified Cache line Clean and Invalidate by Set/Way

DC CIVAC: Data or unified Cache line Clean and Invalidate by VA to PoC

DC CSW: Data or unified Cache line Clean by Set/Way

DC CVAC: Data or unified Cache line Clean by VA to PoC

DC CVADP: Data or unified Cache line Clean by VA to PoDP

DC CVAP: Data or unified Cache line Clean by VA to PoP

DC CVAU: Data or unified Cache line Clean by VA to PoU

DC GVA: Data Cache set Allocation Tag by VA

DC GZVA: Data Cache set Allocation Tags and Zero by VA

DC IGDSW: Invalidate of Data and Allocation Tags by Set/Way

DC IGDVAC: Invalidate of Data and Allocation Tags by VA to PoC

DC IGSW: Invalidate of Allocation Tags by Set/Way

DC IGVAC: Invalidate of Allocation Tags by VA to PoC

DC ISW: Data or unified Cache line Invalidate by Set/Way

DC IVAC: Data or unified Cache line Invalidate by VA to PoC

DC ZVA: Data Cache Zero by VA

DVP RCTX: Data Value Prediction Restriction by Context

IC IALLU: Instruction Cache Invalidate All to PoU

IC IALLUIS: Instruction Cache Invalidate All to PoU, Inner Shareable

IC IVAU: Instruction Cache line Invalidate by VA to PoU

SYS S1 <op1> <Cn> <Cm> <op2>, SYSL S1 <op1> <Cn> <Cm> <op2>, SYSP S1 <op1> <Cn> <Cm> <op2>: IMPLEMENTATION DEFINED maintenance instructions

TLBI ALLE1, TLBI ALLE1NXS: TLB Invalidate All, EL1

TLBI ALLE1IS, TLBI ALLE1ISNXS: TLB Invalidate All, EL1, Inner Shareable

TLBI ALLE1OS, TLBI ALLE1OSNXS: TLB Invalidate All, EL1, Outer Shareable

TLBI ALLE2, TLBI ALLE2NXS: TLB Invalidate All, EL2

TLBI ALLE2IS, TLBI ALLE2ISNXS: TLB Invalidate All, EL2, Inner Shareable

TLBI ALLE2OS, TLBI ALLE2OSNXS: TLB Invalidate All, EL2, Outer Shareable

TLBI ALLE3, TLBI ALLE3NXS: TLB Invalidate All, EL3

TLBI ALLE3IS, TLBI ALLE3ISNXS: TLB Invalidate All, EL3, Inner Shareable

TLBI ALLE3OS, TLBI ALLE3OSNXS: TLB Invalidate All, EL3, Outer Shareable

TLBI ASIDE1, TLBI ASIDE1NXS: TLB Invalidate by ASID, EL1

TLBI ASIDE1IS, TLBI ASIDE1ISNXS: TLB Invalidate by ASID, EL1, Inner Shareable

TLBI ASIDE1OS, TLBI ASIDE1OSNXS: TLB Invalidate by ASID, EL1, Outer Shareable

TLBI IPAS2E1, TLBI IPAS2E1NXS: TLB Invalidate by Intermediate Physical Address, Stage 2, EL1

[TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS](#): TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

[TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS](#): TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

[TLBI IPAS2LE1, TLBI IPAS2LE1NXS](#): TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

[TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS](#): TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

[TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS](#): TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

TLBI PAALL: TLB Invalidate GPT Information by PA, All Entries, Local

TLBI PAALLOS: TLB Invalidate GPT Information by PA, All Entries, Outer Shareable

[TLBI RIPAS2E1, TLBI RIPAS2E1NXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1

[TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

[TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

[TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

[TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

[TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

TLBI RPALOS: TLB Range Invalidate GPT Information by PA, Last level, Outer Shareable

TLBI RPAOS: TLB Range Invalidate GPT Information by PA, Outer Shareable

[TLBI RVAAE1, TLBI RVAAE1NXS](#): TLB Range Invalidate by VA, All ASID, EL1

[TLBI RVAAE1IS, TLBI RVAAE1ISNXS](#): TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

[TLBI RVAAE1OS, TLBI RVAAE1OSNXS](#): TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

[TLBI RVAALE1, TLBI RVAALE1NXS](#): TLB Range Invalidate by VA, All ASID, Last level, EL1

[TLBI RVAALE1IS, TLBI RVAALE1ISNXS](#): TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

[TLBI RVAALE1OS, TLBI RVAALE1OSNXS](#): TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

[TLBI RVAE1, TLBI RVAE1NXS](#): TLB Range Invalidate by VA, EL1

[TLBI RVAE1IS, TLBI RVAE1ISNXS](#): TLB Range Invalidate by VA, EL1, Inner Shareable

[TLBI RVAE1OS, TLBI RVAE1OSNXS](#): TLB Range Invalidate by VA, EL1, Outer Shareable

[TLBI RVAE2, TLBI RVAE2NXS](#): TLB Range Invalidate by VA, EL2

[TLBI RVAE2IS, TLBI RVAE2ISNXS](#): TLB Range Invalidate by VA, EL2, Inner Shareable

[TLBI RVAE2OS, TLBI RVAE2OSNXS](#): TLB Range Invalidate by VA, EL2, Outer Shareable

[TLBI RVAE3, TLBI RVAE3NXS](#): TLB Range Invalidate by VA, EL3

[TLBI RVAE3IS, TLBI RVAE3ISNXS](#): TLB Range Invalidate by VA, EL3, Inner Shareable

[TLBI RVAE3OS, TLBI RVAE3OSNXS](#): TLB Range Invalidate by VA, EL3, Outer Shareable

[TLBI RVALE1, TLBI RVALE1NXS](#): TLB Range Invalidate by VA, Last level, EL1

[TLBI RVALE1IS, TLBI RVALE1ISNXS](#): TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

[TLBI RVALE1OS, TLBI RVALE1OSNXS](#): TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

[TLBI RVALE2, TLBI RVALE2NXS](#): TLB Range Invalidate by VA, Last level, EL2

[TLBI RVALE2IS, TLBI RVALE2ISNXS](#): TLB Range Invalidate by VA, Last level, EL2, Inner Shareable

[TLBI RVALE2OS, TLBI RVALE2OSNXS](#): TLB Range Invalidate by VA, Last level, EL2, Outer Shareable

[TLBI RVALE3, TLBI RVALE3NXS](#): TLB Range Invalidate by VA, Last level, EL3

[TLBI RVALE3IS, TLBI RVALE3ISNXS](#): TLB Range Invalidate by VA, Last level, EL3, Inner Shareable

[TLBI RVALE3OS, TLBI RVALE3OSNXS](#): TLB Range Invalidate by VA, Last level, EL3, Outer Shareable

[TLBI VAAE1, TLBI VAAE1NXS](#): TLB Invalidate by VA, All ASID, EL1

[TLBI VAAE1IS, TLBI VAAE1ISNXS](#): TLB Invalidate by VA, All ASID, EL1, Inner Shareable

[TLBI VAAE1OS, TLBI VAAE1OSNXS](#): TLB Invalidate by VA, All ASID, EL1, Outer Shareable

[TLBI VAALE1, TLBI VAALE1NXS](#): TLB Invalidate by VA, All ASID, Last level, EL1

[TLBI VAALE1IS, TLBI VAALE1ISNXS](#): TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

[TLBI VAALE1OS, TLBI VAALE1OSNXS](#): TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

[TLBI VAE1, TLBI VAE1NXS](#): TLB Invalidate by VA, EL1

[TLBI VAE1IS, TLBI VAE1ISNXS](#): TLB Invalidate by VA, EL1, Inner Shareable

[TLBI VAE1OS, TLBI VAE1OSNXS](#): TLB Invalidate by VA, EL1, Outer Shareable

[TLBI VAE2, TLBI VAE2NXS](#): TLB Invalidate by VA, EL2

[TLBI VAE2IS, TLBI VAE2ISNXS](#): TLB Invalidate by VA, EL2, Inner Shareable

[TLBI VAE2OS, TLBI VAE2OSNXS](#): TLB Invalidate by VA, EL2, Outer Shareable

[TLBI VAE3, TLBI VAE3NXS](#): TLB Invalidate by VA, EL3

[TLBI VAE3IS, TLBI VAE3ISNXS](#): TLB Invalidate by VA, EL3, Inner Shareable

[TLBI VAE3OS, TLBI VAE3OSNXS](#): TLB Invalidate by VA, EL3, Outer Shareable

[TLBI VALE1, TLBI VALE1NXS](#): TLB Invalidate by VA, Last level, EL1

[TLBI VALE1IS, TLBI VALE1ISNXS](#): TLB Invalidate by VA, Last level, EL1, Inner Shareable

[TLBI VALE1OS, TLBI VALE1OSNXS](#): TLB Invalidate by VA, Last level, EL1, Outer Shareable

[TLBI VALE2, TLBI VALE2NXS](#): TLB Invalidate by VA, Last level, EL2

[TLBI VALE2IS, TLBI VALE2ISNXS](#): TLB Invalidate by VA, Last level, EL2, Inner Shareable

[TLBI VALE2OS, TLBI VALE2OSNXS](#): TLB Invalidate by VA, Last level, EL2, Outer Shareable

[TLBI VALE3, TLBI VALE3NXS](#): TLB Invalidate by VA, Last level, EL3

[TLBI VALE3IS, TLBI VALE3ISNXS](#): TLB Invalidate by VA, Last level, EL3, Inner Shareable

[TLBI VALE3OS, TLBI VALE3OSNXS](#): TLB Invalidate by VA, Last level, EL3, Outer Shareable

[TLBI VMALLE1, TLBI VMALLE1NXS](#): TLB Invalidate by VMID, All at stage 1, EL1

[TLBI VMALLE1IS, TLBI VMALLE1ISNXS](#): TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

[TLBI VMALLE1OS, TLBI VMALLE1OSNXS](#): TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable

[TLBI VMALLS12E1, TLBI VMALLS12E1NXS](#): TLB Invalidate by VMID, All at Stage 1 and 2, EL1

[TLBI VMALLS12E1IS, TLBI VMALLS12E1ISNXS](#): TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

TLBI VMALLS12E1OS, TLBI VMALLS12E1OSNXS: TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable

[TLBIP IPAS2E1, TLBIP IPAS2E1NXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1

[TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

[TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

[TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1

[TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

[TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS](#): TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

[TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1

[TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

[TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

[TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

[TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

[TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS](#): TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

[TLBIP RVAAE1, TLBIP RVAAE1NXS](#): TLB Range Invalidate by VA, All ASID, EL1

[TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS](#): TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

[TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS](#): TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

[TLBIP RVAAL1, TLBIP RVAAL1NXS](#): TLB Range Invalidate by VA, All ASID, Last level, EL1

[TLBIP RVAAL1IS, TLBIP RVAAL1ISNXS](#): TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

[TLBIP RVAAL1OS, TLBIP RVAAL1OSNXS](#): TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

[TLBIP RVAE1, TLBIP RVAE1NXS](#): TLB Range Invalidate by VA, EL1

[TLBIP RVAE1IS, TLBIP RVAE1ISNXS](#): TLB Range Invalidate by VA, EL1, Inner Shareable

[TLBIP RVAE1OS, TLBIP RVAE1OSNXS](#): TLB Range Invalidate by VA, EL1, Outer Shareable

[TLBIP RVAE2, TLBIP RVAE2NXS](#): TLB Range Invalidate by VA, EL2

[TLBIP RVAE2IS, TLBIP RVAE2ISNXS](#): TLB Range Invalidate by VA, EL2, Inner Shareable

[TLBIP RVAE2OS, TLBIP RVAE2OSNXS](#): TLB Range Invalidate by VA, EL2, Outer Shareable

[TLBIP RVAE3, TLBIP RVAE3NXS](#): TLB Range Invalidate by VA, EL3

[TLBIP RVAE3IS, TLBIP RVAE3ISNXS](#): TLB Range Invalidate by VA, EL3, Inner Shareable

[TLBIP RVAE3OS, TLBIP RVAE3OSNXS](#): TLB Range Invalidate by VA, EL3, Outer Shareable

[TLBIP RVALE1, TLBIP RVALE1NXS](#): TLB Range Invalidate by VA, Last level, EL1

[TLBIP RVALE1IS, TLBIP RVALE1ISNXS](#): TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

[TLBIP RVALE1OS, TLBIP RVALE1OSNXS](#): TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

[TLBIP RVALE2, TLBIP RVALE2NXS](#): TLB Range Invalidate by VA, Last level, EL2

[TLBIP RVALE2IS, TLBIP RVALE2ISNXS](#): TLB Range Invalidate by VA, Last level, EL2, Inner Shareable

[TLBIP RVALE2OS, TLBIP RVALE2OSNXS](#): TLB Range Invalidate by VA, Last level, EL2, Outer Shareable

[TLBIP RVALE3, TLBIP RVALE3NXS](#): TLB Range Invalidate by VA, Last level, EL3

[TLBIP RVALE3IS, TLBIP RVALE3ISNXS](#): TLB Range Invalidate by VA, Last level, EL3, Inner Shareable

[TLBIP RVALE3OS, TLBIP RVALE3OSNXS](#): TLB Range Invalidate by VA, Last level, EL3, Outer Shareable

[TLBIP VAAE1, TLBIP VAAE1NXS](#): TLB Invalidate Pair by VA, All ASID, EL1

[TLBIP VAAE1IS, TLBIP VAAE1ISNXS](#): TLB Invalidate Pair by VA, All ASID, EL1, Inner Shareable

[TLBIP VAAE1OS, TLBIP VAAE1OSNXS](#): TLB Invalidate Pair by VA, All ASID, EL1, Outer Shareable

[TLBIP VAALE1, TLBIP VAALE1NXS](#): TLB Invalidate Pair by VA, All ASID, Last level, EL1

[TLBIP VAALE1IS, TLBIP VAALE1ISNXS](#): TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Inner Shareable

[TLBIP VAALE1OS, TLBIP VAALE1OSNXS](#): TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Outer Shareable

[TLBIP VAE1, TLBIP VAE1NXS](#): TLB Invalidate Pair by VA, EL1

[TLBIP VAE1IS, TLBIP VAE1ISNXS](#): TLB Invalidate Pair by VA, EL1, Inner Shareable

[TLBIP VAE1OS, TLBIP VAE1OSNXS](#): TLB Invalidate Pair by VA, EL1, Outer Shareable

[TLBIP VAE2, TLBIP VAE2NXS](#): TLB Invalidate Pair by VA, EL2

[TLBIP VAE2IS, TLBIP VAE2ISNXS](#): TLB Invalidate Pair by VA, EL2, Inner Shareable

[TLBIP VAE2OS, TLBIP VAE2OSNXS](#): TLB Invalidate Pair by VA, EL2, Outer Shareable

[TLBIP VAE3, TLBIP VAE3NXS](#): TLB Invalidate Pair by VA, EL3

[TLBIP VAE3IS, TLBIP VAE3ISNXS](#): TLB Invalidate Pair by VA, EL3, Inner Shareable

[TLBIP VAE3OS, TLBIP VAE3OSNXS](#): TLB Invalidate Pair by VA, EL3, Outer Shareable

[TLBIP VALE1, TLBIP VALE1NXS](#): TLB Invalidate Pair by VA, Last level, EL1

[TLBIP VALE1IS, TLBIP VALE1ISNXS](#): TLB Invalidate Pair by VA, Last level, EL1, Inner Shareable

[TLBIP VALE1OS, TLBIP VALE1OSNXS](#): TLB Invalidate Pair by VA, Last level, EL1, Outer Shareable

[TLBIP VALE2, TLBIP VALE2NXS](#): TLB Invalidate Pair by VA, Last level, EL2

[TLBIP VALE2IS, TLBIP VALE2ISNXS](#): TLB Invalidate Pair by VA, Last level, EL2, Inner Shareable

[TLBIP VALE2OS, TLBIP VALE2OSNXS](#): TLB Invalidate Pair by VA, Last level, EL2, Outer Shareable

[TLBIP VALE3, TLBIP VALE3NXS](#): TLB Invalidate Pair by VA, Last level, EL3

[TLBIP VALE3IS, TLBIP VALE3ISNXS](#): TLB Invalidate Pair by VA, Last level, EL3, Inner Shareable

[TLBIP VALE3OS, TLBIP VALE3OSNXS](#): TLB Invalidate Pair by VA, Last level, EL3, Outer Shareable

[TRCIT](#): Trace Instrumentation

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ADEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nACCDATA_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ACCDATA_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ADEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ACCDATA_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACCDATA_EL1;

```

MSR ACCDATA_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ADEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nACCDATA_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ACCDATA_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ADEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ACCDATA_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACCDATA_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)

The AFSR0_EL1 characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL1.

Configuration

AArch64 System register AFSR0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ADESR\[31:0\]](#).

Attributes

AFSR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AFSR0_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR0_EL1 or AFSR0_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AFSR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x128];
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;

```

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];

```

MRS <Xt>, AFSR0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x128];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;

```

MSR AFSR0_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x128] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

The AFSR0_EL2 characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL2.

Configuration

AArch64 System register AFSR0_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HADFSTR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

AFSR0_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AFSR0_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR0_EL2 or AFSR0_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AFSR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL2;

```

MSR AFSR0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL2 = X[t, 64];

```

MRS <Xt>, AFSR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x128];
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;

```

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)

The AFSR1_EL1 characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL1.

Configuration

AArch64 System register AFSR1_EL1 bits [31:0] are architecturally mapped to AArch32 System register [AIFSR\[31:0\]](#).

Attributes

AFSR1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AFSR1_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR1_EL1 or AFSR1_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AFSR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x130];
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;

```

MSR AFSR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t, 64];

```

MRS <Xt>, AFSR1_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x130];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;

```

MSR AFSR1_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x130] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

The AFSR1_EL2 characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL2.

Configuration

AArch64 System register AFSR1_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HIAFSR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

AFSR1_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AFSR1_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AFSR1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL2;

```

MSR AFSR1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t, 64];

```

MRS <Xt>, AFSR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x130];
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;

```

MSR AFSR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            AFSR1_EL2 = X[t, 64];
        else
            AFSR1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        AFSR1_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AIDR_EL1, Auxiliary ID Register

The AIDR_EL1 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED identification information.

The value of this register must be interpreted in conjunction with the value of [MIDR_EL1](#).

Configuration

AArch64 System register AIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [AIDR\[31:0\]](#).

Attributes

AIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing AIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b111

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
        && HFGTR_EL2.AIDR_EL1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = AIDR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = AIDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AIDR_EL1;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AMAIR2_EL1, Extended Auxiliary Memory Attribute Indirection Register (EL1)

The AMAIR2_EL1 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR2_EL1](#).

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to AMAIR2_EL1 are UNDEFINED.

Attributes

AMAIR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AMAIR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMAIR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nAMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x288];
    else
        X[t, 64] = AMAIR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR2_EL2;
    else
        X[t, 64] = AMAIR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR2_EL1;

```

MSR AMAIR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nAMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x288] = X[t, 64];
    else
        AMAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        AMAIR2_EL2 = X[t, 64];
    else
        AMAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR2_EL1 = X[t, 64];

```

MRS <Xt>, AMAIR2_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x288];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.AIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMAIR2_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR2_EL1;
    else
        UNDEFINED;

```

MSR AMAIR2_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x288] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.AIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AMAIR2_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        AMAIR2_EL1 = X[t, 64];
    else
        UNDEFINED;

```


no old file	htmldiff from-	(new)
-------------	----------------	-------

AMAIR2_EL2, Extended Auxiliary Memory Attribute Indirection Register (EL2)

The AMAIR2_EL2 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR2_EL2](#).

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to AMAIR2_EL2 are UNDEFINED.

Attributes

AMAIR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing AMAIR2_EL2

When FEAT_VHE is implemented, and [HCR_EL2](#).E2H is 1, without explicit synchronization, accesses from EL2 using the register name AMAIR2_EL2 or AMAIR2_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMAIR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMAIR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR2_EL2;

```

MSR AMAIR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AMAIR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR2_EL2 = X[t, 64];

```

MRS <Xt>, AMAIR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nAMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x288];
    else
        X[t, 64] = AMAIR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR2_EL2;
    else
        X[t, 64] = AMAIR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR2_EL1;
    
```

MSR AMAIR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nAMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x288] = X[t, 64];
    else
        AMAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        AMAIR2_EL2 = X[t, 64];
    else
        AMAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR2_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

AMAIR2_EL3, Extended Auxiliary Memory Attribute Indirection Register (EL3)

The AMAIR2_EL3 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR2_EL3](#).

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to AMAIR2_EL3 are UNDEFINED.

Attributes

AMAIR2_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing AMAIR2_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMAIR2_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR2_EL3;
```

MSR AMAIR2_EL3, <Xt>

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b110	0b1010	0b0011	0b001
------	-------	--------	--------	-------

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AMAIR2_EL3 = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

The AMAIR_EL1 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR_EL1](#).

Configuration

AArch64 System register AMAIR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [AMAIRO\[31:0\]](#).

AArch64 System register AMAIR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [AMAIR1\[31:0\]](#).

Attributes

AMAIR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

AMAIR_EL1 is permitted to be cached in a TLB.

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AMAIR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AMAIR_EL1 or AMAIR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x148];
    else
        X[t, 64] = AMAIR_EL1;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            X[t, 64] = AMAIR_EL2;
        else
            X[t, 64] = AMAIR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMAIR_EL1;
    
```

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            AMAIR_EL2 = X[t, 64];
        else
            AMAIR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        AMAIR_EL1 = X[t, 64];
    
```

MRS <Xt>, AMAIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x148];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;

```

MSR AMAIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x148] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

The AMAIR_EL2 characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR_EL2](#).

Configuration

AArch64 System register AMAIR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HMAIR0\[31:0\]](#).

AArch64 System register AMAIR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HMAIR1\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

AMAIR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

AMAIR_EL2 is permitted to be cached in a TLB.

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AMAIR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMAIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AMAIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL2;

```

MSR AMAIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL2 = X[t, 64];

```

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x148];
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL2;
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL1;

```

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMCNTENCLR0_EL0, Activity Monitors Count Enable Clear Register 0

The AMCNTENCLR0_EL0 characteristics are:

Purpose

Disable control bits for the architected activity monitors event counters, [AMEVCNTR0<n>_EL0](#).

Configuration

AArch64 System register AMCNTENCLR0_EL0 bits [31:0] are architecturally mapped to AArch32 System register [AMCNTENCLR0\[31:0\]](#).

AArch64 System register AMCNTENCLR0_EL0 bits [31:0] are architecturally mapped to External register [AMCNTENCLR0\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENCLR0_EL0 are UNDEFINED.

Attributes

AMCNTENCLR0_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RAZ/WI															
RES0																RAZ/WI															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

Bits [15:4]

Reserved, RAZ/WI.

This field is reserved for additional architected activity monitor event counters, which Arm might define in a future version of the Activity Monitors architecture.

P<n>, bit [n], for n = 3 to 0

Activity monitor event counter disable bit for [AMEVCNTR0<n>_EL0](#).

Note

[AMCGCR_EL0](#).CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4.

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR0<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR0<n>_EL0 is enabled. When written, disables AMEVCNTR0<n>_EL0 .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENCLR0_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMCNTENCLR0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR0_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMCNTEN0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR0_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR0_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMCNTENCLR0_EL0;

```

MSR AMCNTENCLR0_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b100

```

if IsHighestEL(PSTATE.EL) then
    AMCNTENCLR0_EL0 = X[t, 64];
else
    UNDEFINED;

```


(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

AMCNTENCLR1_EL0, Activity Monitors Count Enable Clear Register 1

The AMCNTENCLR1_EL0 characteristics are:

Purpose

Disable control bits for the auxiliary activity monitors event counters, [AMEVCNTR1<n>_EL0](#).

Configuration

AArch64 System register AMCNTENCLR1_EL0 bits [31:0] are architecturally mapped to AArch32 System register [AMCNTENCLR1\[31:0\]](#).

AArch64 System register AMCNTENCLR1_EL0 bits [31:0] are architecturally mapped to External register [AMCNTENCLR1\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENCLR1_EL0 are UNDEFINED.

Attributes

AMCNTENCLR1_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

P<n>, bit [n], for n = 15 to 0

Activity monitor event counter disable bit for [AMEVCNTR1<n>_EL0](#).

When N is less than 16, bits [15:N] are RAZ/WI, where N is the value in [AMCGCR_EL0.CG1NC](#).

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR1<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR1<n>_EL0 is enabled. When written, disables AMEVCNTR1<n>_EL0 .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENCLR1_EL0

If the number of auxiliary activity monitor event counters implemented is zero, reads and writes of AMCNTENCLR1_EL0 are UNDEFINED.

Note

The number of auxiliary activity monitor event counters implemented is zero exactly when [AMCFGR_EL0](#).NCG == 0b0000.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMCNTENCLR1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR1_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMCNTEN1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR1_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENCLR1_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMCNTENCLR1_EL0;

```

MSR AMCNTENCLR1_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0011	0b000

```

if IsHighestEL(PSTATE.EL) then
    AMCNTENCLR1_EL0 = X[t, 64];
else
    UNDEFINED;

```

(old)	htmldiff from-	(new)
-------	----------------	-------

AMCNTENSET0_EL0, Activity Monitors Count Enable Set Register 0

The AMCNTENSET0_EL0 characteristics are:

Purpose

Enable control bits for the architected activity monitors event counters, [AMEVCNTR0<n>_EL0](#).

Configuration

AArch64 System register AMCNTENSET0_EL0 bits [31:0] are architecturally mapped to AArch32 System register [AMCNTENSET0\[31:0\]](#).

AArch64 System register AMCNTENSET0_EL0 bits [31:0] are architecturally mapped to External register [AMCNTENSET0\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENSET0_EL0 are UNDEFINED.

Attributes

AMCNTENSET0_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																RAZ/WI															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

Bits [15:4]

Reserved, RAZ/WI.

This field is reserved for additional architected activity monitor event counters, which Arm might define in a future version of the Activity Monitors architecture.

P<n>, bit [n], for n = 3 to 0

Activity monitor event counter enable bit for [AMEVCNTR0<n>_EL0](#).

Note

[AMCGCR_EL0](#).CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4.

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR0<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR0<n>_EL0 is enabled. When written, enables AMEVCNTR0<n>_EL0 .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENSET0_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMCNTENSET0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET0_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMCNTEN0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET0_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET0_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMCNTENSET0_EL0;

```

MSR AMCNTENSET0_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b101

```

if IsHighestEL(PSTATE.EL) then
    AMCNTENSET0_EL0 = X[t, 64];
else
    UNDEFINED;

```


(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

AMCNTENSET1_EL0, Activity Monitors Count Enable Set Register 1

The AMCNTENSET1_EL0 characteristics are:

Purpose

Enable control bits for the auxiliary activity monitors event counters, [AMEVCNTR1<n>_EL0](#).

Configuration

AArch64 System register AMCNTENSET1_EL0 bits [31:0] are architecturally mapped to AArch32 System register [AMCNTENSET1\[31:0\]](#).

AArch64 System register AMCNTENSET1_EL0 bits [31:0] are architecturally mapped to External register [AMCNTENSET1\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENSET1_EL0 are UNDEFINED.

Attributes

AMCNTENSET1_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

P<n>, bit [n], for n = 15 to 0

Activity monitor event counter enable bit for [AMEVCNTR1<n>_EL0](#).

When N is less than 16, bits [15:N] are RAZ/WI, where N is the value in [AMCGCR_EL0.CG1NC](#).

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR1<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR1<n>_EL0 is enabled. When written, enables AMEVCNTR1<n>_EL0 .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENSET1_EL0

If the number of auxiliary activity monitor event counters implemented is zero, reads and writes of AMCNTENSET1_EL0 are UNDEFINED.

Note

The number of auxiliary activity monitor counters implemented is zero when [AMCFGR_EL0](#).NCG == 0b0000.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMCNTENSET1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0011	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET1_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMCNTEN1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET1_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMCNTENSET1_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMCNTENSET1_EL0;

```

MSR AMCNTENSET1_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0011	0b001

```

if IsHighestEL(PSTATE.EL) then
    AMCNTENSET1_EL0 = X[t, 64];
else
    UNDEFINED;

```

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

AMEVCNTR0<n>_EL0, Activity Monitors Event Counter Registers 0, n = 0 - 3

The AMEVCNTR0<n>_EL0 characteristics are:

Purpose

Provides access to the architected activity monitor event counters.

Configuration

AArch64 System register AMEVCNTR0<n>_EL0 bits [63:0] are architecturally mapped to AArch32 System register [AMEVCNTR0<n>\[63:0\]](#).

AArch64 System register AMEVCNTR0<n>_EL0 bits [63:0] are architecturally mapped to External register [AMEVCNTR0<n>\[63:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVCNTR0<n>_EL0 are UNDEFINED.

Attributes

AMEVCNTR0<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ACNT															
																ACNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ACNT, bits [63:0]

Architected activity monitor event counter n.

Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.

If FEAT_AMUv1p1 is implemented, [HCR_EL2](#).AMVOFFEN is 1, [SCR_EL3](#).AMVOFFEN is 1, [HCR_EL2](#).{E2H, TGE} is not {1,1}, and EL2 is implemented in the current Security state, access to these registers at EL0 or EL1 return (PCount<63:0> - [AMEVCNTVOFF0<n>_EL2](#)<63:0>).

PCount is the physical count returned when AMEVCNTR0<n>_EL0 is read from EL2 or EL3.

If the counter is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMEVCNTR0<n>_EL0

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are UNDEFINED.

Note

[AMCGCR_EL0](#).CG0NC identifies the number of architected activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMEVCNTR0<m>_EL0 ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b010:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if m >= 4 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMEVCNTR0<m>_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[m];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMEVCNTR0<m>_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR0_EL0[m];
```

MSR AMEVCNTR0<m>_EL0, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b010:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);
```

```
if m >= 4 then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR0_EL0[m] = X[t, 64];
else
    UNDEFINED;
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMEVCNTR1<n>_EL0, Activity Monitors Event Counter Registers 1, n = 0 - 15

The AMEVCNTR1<n>_EL0 characteristics are:

Purpose

Provides access to the auxiliary activity monitor event counters.

Configuration

AArch64 System register AMEVCNTR1<n>_EL0 bits [63:0] are architecturally mapped to AArch32 System register [AMEVCNTR1<n>\[63:0\]](#).

AArch64 System register AMEVCNTR1<n>_EL0 bits [63:0] are architecturally mapped to External register [AMEVCNTR1<n>\[63:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVCNTR1<n>_EL0 are UNDEFINED.

Attributes

AMEVCNTR1<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ACNT															
																ACNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ACNT, bits [63:0]

Auxiliary activity monitor event counter n.

Value of auxiliary activity monitor event counter n, where n is the number of this register and is a number from 0 to 15.

If FEAT_AMUv1p1 is implemented, [HCR_EL2](#).AMVOFFEN is 1, [SCR_EL3](#).AMVOFFEN is 1, [HCR_EL2](#).{E2H, TGE} is not {1,1}, EL2 is implemented in the current Security state, and [AMCR_EL0](#).CG1RZ is 0, reads to these registers at EL0 or EL1 return (PCount<63:0> - [AMEVCNTVOFF1<n>_EL2<63:0>](#)).

PCount is the physical count returned when AMEVCNTR1<n>_EL0 is read from EL2 or EL3.

If the counter is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMEVCNTR1<n>_EL0

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are UNDEFINED.

Note

[AMCGCR_EL0](#).CG1NC identifies the number of auxiliary activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMEVCNTR1<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b110:m[3]	m[2:0]

```

integer m = UInt(CRm<0>:op2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMEVCNTR1<m>_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif AMCR_EL0.CG1RZ == '1' then
            X[t, 64] = Zeros(64);
        else
            X[t, 64] = AMEVCNTR1_EL0[m];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMEVCNTR1<m>_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif !IsHighestEL(PSTATE.EL) && AMCR_EL0.CG1RZ == '1' then
            X[t, 64] = Zeros(64);
        else
            X[t, 64] = AMEVCNTR1_EL0[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif !IsHighestEL(PSTATE.EL) && AMCR_EL0.CG1RZ == '1' then
            X[t, 64] = Zeros(64);
        else
            X[t, 64] = AMEVCNTR1_EL0[m];
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMEVCNTR1_EL0[m];

```

MSR AMEVCNTR1<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b110:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR1_EL0[m] = X[t, 64];
else
    UNDEFINED;
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMEVTYPER1<n>_EL0, Activity Monitors Event Type Registers 1, n = 0 - 15

The AMEVTYPER1<n>_EL0 characteristics are:

Purpose

Provides information on the events that an auxiliary activity monitor event counter [AMEVCNTR1<n>_EL0](#) counts.

Configuration

AArch64 System register AMEVTYPER1<n>_EL0 bits [31:0] are architecturally mapped to AArch32 System register [AMEVTYPER1<n>\[31:0\]](#).

AArch64 System register AMEVTYPER1<n>_EL0 bits [31:0] are architecturally mapped to External register [AMEVTYPER1<n>\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVTYPER1<n>_EL0 are UNDEFINED.

Attributes

AMEVTYPER1<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																																
RES0																evtCount																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:16]

Reserved, RES0.

evtCount, bits [15:0]

Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter [AMEVCNTR1<n>_EL0](#).

It is IMPLEMENTATION DEFINED what values are supported by each counter.

If software writes a value to this field which is not supported by the corresponding counter [AMEVCNTR1<n>_EL0](#), then:

- It is UNPREDICTABLE which event will be counted.
- The value read back is UNKNOWN.

The event counted by [AMEVCNTR1<n>_EL0](#) might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.

If the corresponding counter [AMEVCNTR1<n>_EL0](#) is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AMEVTYPER1<n>_EL0

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.

Note

[AMCGCR_EL0](#).CG1NC identifies the number of auxiliary activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, AMEVTYPER1<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b111:m[3]	m[2:0]

```

integer m = UInt(CRm<0>:op2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMEVTYPER1<m>_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMEVTYPER1_EL0[m];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HAFGRTR_EL2.AMEVTYPER1<m>_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMEVTYPER1_EL0[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TAM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TAM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = AMEVTYPER1_EL0[m];
    elsif PSTATE.EL == EL3 then
        X[t, 64] = AMEVTYPER1_EL0[m];

```

MSR AMEVTYPER1<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b111:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) && !boolean IMPLEMENTATION_DEFINED "AMEVCNTR1_EL0[m] is fixed" then
    AMEVTYPER1_EL0[m] = X[t, 64];
else
    UNDEFINED;
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

APDAKeyHi_EL1, Pointer Authentication Key A for Data (bits[127:64])

The APDAKeyHi_EL1 characteristics are:

Purpose

Holds bits[127:64] of key A used for authentication of data pointer values.

Note

The term APDAKey_EL1 is used to describe the concatenation of [APDAKeyHi_EL1](#): [APDAKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APDAKeyHi_EL1 are UNDEFINED.

Attributes

APDAKeyHi_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

64 bit value, bits[127:64] of the 128 bit pointer authentication key value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APDAKeyHi_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APDAKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APDAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDAKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDAKeyHi_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APDAKeyHi_EL1;

```

MSR APDAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APDAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDAKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDAKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APDAKeyHi_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APDAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDAKeyLo_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDAKeyLo_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APDAKeyLo_EL1;

```

MSR APDAKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APDAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDAKeyLo_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                APDAKeyLo_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        APDAKeyLo_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

APDBKeyHi_EL1, Pointer Authentication Key B for Data (bits[127:64])

The APDBKeyHi_EL1 characteristics are:

Purpose

Holds bits[127:64] of key B used for authentication of data pointer values.

Note

The term APDBKey_EL1 is used to describe the concatenation of [APDBKeyHi_EL1](#): [APDBKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APDBKeyHi_EL1 are UNDEFINED.

Attributes

APDBKeyHi_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

64 bit value, bits[127:64] of the 128 bit pointer authentication key value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APDBKeyHi_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APDBKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APDBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDBKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDBKeyHi_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APDBKeyHi_EL1;

```

MSR APDBKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APDBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDBKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDBKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APDBKeyHi_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APDBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDBKeyLo_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APDBKeyLo_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APDBKeyLo_EL1;

```

MSR APDBKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APDBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDBKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APDBKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APDBKeyLo_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

APGAKeyHi_EL1, Pointer Authentication Key A for Code (bits[127:64])

The APGAKeyHi_EL1 characteristics are:

Purpose

Holds bits[127:64] of key used for generic pointer authentication code.

Note

The term APGAKey_EL1 is used to describe the concatenation of [APGAKeyHi_EL1](#): [APGAKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APGAKeyHi_EL1 are UNDEFINED.

Attributes

APGAKeyHi_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
64 bit value, bits[127:64] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

64 bit value, bits[127:64] of the 128 bit pointer authentication key value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APGAKeyHi_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APGAKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APGAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APGAKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APGAKeyHi_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APGAKeyHi_EL1;

```

MSR APGAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APGAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APGAKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APGAKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APGAKeyHi_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

APGAKeyLo_EL1, Pointer Authentication Key A for Code (bits[63:0])

The APGAKeyLo_EL1 characteristics are:

Purpose

Holds bits[63:0] of key used for generic pointer authentication code.

Note

The term APGAKey_EL1 is used to describe the concatenation of [APGAKeyHi_EL1](#): [APGAKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APGAKeyLo_EL1 are UNDEFINED.

Attributes

APGAKeyLo_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APGAKeyLo_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APGAKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APGAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APGAKeyLo_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APGAKeyLo_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APGAKeyLo_EL1;

```

MSR APGAKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APGAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APGAKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APGAKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APGAKeyLo_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APIAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIAKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIAKeyHi_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APIAKeyHi_EL1;

```

MSR APIAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APIAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIAKeyHi_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                APIAKeyHi_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        APIAKeyHi_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

APIAKeyLo_EL1, Pointer Authentication Key A for Instruction (bits[63:0])

The APIAKeyLo_EL1 characteristics are:

Purpose

Holds bits[63:0] of key A used for authentication of instruction pointer values.

Note

The term APIAKey_EL1 is used to describe the concatenation of [APIAKeyHi_EL1](#): [APIAKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APIAKeyLo_EL1 are UNDEFINED.

Attributes

APIAKeyLo_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APIAKeyLo_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APIAKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APIAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIAKeyLo_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIAKeyLo_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APIAKeyLo_EL1;

```

MSR APIAKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APIAKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIAKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIAKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APIAKeyLo_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APIBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIBKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIBKeyHi_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APIBKeyHi_EL1;

```

MSR APIBKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APIBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIBKeyHi_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                APIBKeyHi_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        APIBKeyHi_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

APIBKeyLo_EL1, Pointer Authentication Key B for Instruction (bits[63:0])

The APIBKeyLo_EL1 characteristics are:

Purpose

Holds bits[63:0] of key B used for authentication of instruction pointer values.

Note

The term APIBKey_EL1 is used to describe the concatenation of [APIBKeyHi_EL1](#): [APIBKeyLo_EL1](#).

Configuration

This register is present only when FEAT_PAuth is implemented. Otherwise, direct accesses to APIBKeyLo_EL1 are UNDEFINED.

Attributes

APIBKeyLo_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
64 bit value, bits[63:0] of the 128 bit pointer authentication key value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

- 64 bit value, bits[63:0] of the 128 bit pointer authentication key value.
- The reset behavior of this field is:
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing APIBKeyLo_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, APIBKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.APIBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIBKeyLo_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = APIBKeyLo_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = APIBKeyLo_EL1;

```

MSR APIBKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.APIBKey == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIBKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.APK == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.APK == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            APIBKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    APIBKeyLo_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E0R, Address Translate Stage 1 EL0 Read

The AT S1E0R characteristics are:

Purpose

Performs stage 1 address translation from EL0, with permissions as if reading from the given virtual address from EL0, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3.NS](#):
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.
- Otherwise, the EL1&0 translation regime.

Configuration

There are no configuration notes.

Attributes

AT S1E0R is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E0R instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E0R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E0R == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AT S1E0W, Address Translate Stage 1 EL0 Write

The AT S1E0W characteristics are:

Purpose

Performs stage 1 address translation from EL0, with permissions as if writing to the given virtual address from EL0, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3.NS](#):
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.
- Otherwise, the EL1&0 translation regime.

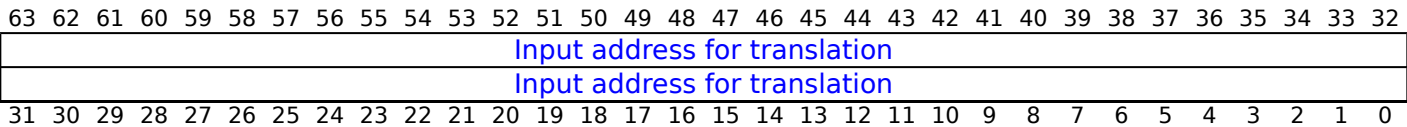
Configuration

There are no configuration notes.

Attributes

AT S1E0W is a 64-bit System instruction.

Field descriptions



Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E0W instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E0W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E0W == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E1R, Address Translate Stage 1 EL1 Read

The AT S1E1R characteristics are:

Purpose

Performs stage 1 address translation, with permissions as if reading from the given virtual address from EL1, or from EL2 if the Effective value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3](#).NS:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime, accessed from EL1.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime, accessed from EL2.
- Otherwise, the EL1&0 translation regime, accessed from EL1.

Configuration

There are no configuration notes.

Attributes

AT S1E1R is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E1R instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E1R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E1R == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E1RP, Address Translate Stage 1 EL1 Read PAN

The AT S1E1RP characteristics are:

Purpose

Performs a stage 1 address translation, where the value of PSTATE.PAN determines if a read from a location will generate a Permission fault for a privileged access, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3.NS](#):
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime, accessed from EL1.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime, accessed from EL2.
- Otherwise, the EL1&0 translation regime, accessed from EL1.

Configuration

This instruction is present only when FEAT_PAN2 is implemented. Otherwise, direct accesses to AT S1E1RP are UNDEFINED.

Attributes

AT S1E1RP is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E1RP instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E1RP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E1RP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_ReadPAN);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_ReadPAN);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_ReadPAN);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AT S1E1W, Address Translate Stage 1 EL1 Write

The AT S1E1W characteristics are:

Purpose

Performs stage 1 address translation, with permissions as if writing to the given virtual address from EL1, or from EL2 if the Effective value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3](#).NS:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime, accessed from EL1.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime, accessed from EL2.
- Otherwise, the EL1&0 translation regime, accessed from EL1.

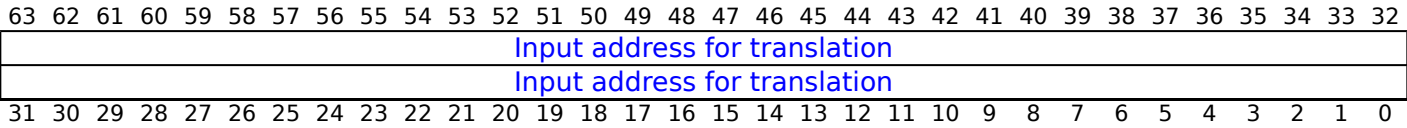
Configuration

There are no configuration notes.

Attributes

AT S1E1W is a 64-bit System instruction.

Field descriptions



Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E1W instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E1W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E1W == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E1WP, Address Translate Stage 1 EL1 Write PAN

The AT S1E1WP characteristics are:

Purpose

Performs a stage 1 address translation, where the value of PSTATE.PAN determines if a write to a location will generate a Permission fault for a privileged access, using the following translation regime:

- When EL2 is implemented and enabled in the Security state described by the current value of [SCR_EL3.NS](#):
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime, accessed from EL1.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime, accessed from EL2.
- Otherwise, the EL1&0 translation regime, accessed from EL1.

Configuration

This instruction is present only when FEAT_PAN2 is implemented. Otherwise, direct accesses to AT S1E1WP are UNDEFINED.

Attributes

AT S1E1WP is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

Executing the AT S1E1WP instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

AT S1E1WP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.AT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ATS1E1WP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_WritePAN);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_WritePAN);
elsif PSTATE.EL == EL3 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_WritePAN);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRB IALL, Invalidate the Branch Record Buffer

The BRB IALL characteristics are:

Purpose

Invalidates all Branch records in the Branch Record Buffer.

Configuration

This instruction is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRB IALL are UNDEFINED.

Attributes

BRB IALL is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the BRB IALL instruction

Rt should be encoded as 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

BRB IALL

op0	op1	CRn	CRm	op2
0b01	0b001	0b0111	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.nBRBIALL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRB_IALL();
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRB_IALL();
elsif PSTATE.EL == EL3 then
    BRB_IALL();

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

BRB INJ, Branch Record Injection into the Branch Record Buffer

The BRB INJ characteristics are:

Purpose

Injects the Branch Record held in [BRBINFINJ_EL1](#), [BRBSRCINJ_EL1](#), and [BRBTGTINJ_EL1](#) into the Branch Record Buffer.

Configuration

This instruction is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRB INJ are UNDEFINED.

Attributes

BRB INJ is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the BRB INJ instruction

Rt should be encoded as 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

BRB INJ

op0	op1	CRn	CRm	op2
0b01	0b001	0b0111	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.nBRBINJ == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRB_INJ();
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRB_INJ();
elsif PSTATE.EL == EL3 then
    BRB_INJ();

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ERTN	Meaning
0b0	Disable the recording Branch records for exception return instructions from EL1.
0b1	Enable the recording Branch records for exception return instructions from EL1.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Bits [21:109]

Reserved, RES0.

FZPSS, bit [9]

When FEAT_PMUv3_SS is implemented:

Freeze BRBE on PMU Snapshot.

FZPSS	Meaning
0b0	Branch recording is not affected by this control.
0b1	If either EL2 is not implemented or BRBCR_EL2.FZPSS is 1, then a BRBE freeze event occurs when a PMU snapshot occurs.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FZP, bit [8]

When FEAT_PMUv3 is implemented:

Freeze BRBE on PMU overflow.

FZP	Meaning
0b0	Branch recording is not affected by this control.
0b1	A BRBE freeze event occurs when a PMU overflow occurs.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [7]

Reserved, RES0.

TS, bits [6:5]

Timestamp Control.

TS	Meaning	Applies when
0b01	Virtual timestamp. The BRBE recorded timestamp is the physical counter value, minus the value of CNTVOFF_EL2 .	When FEAT_ECV is implemented
0b10	Guest physical timestamp. The BRBE recorded timestamp is the physical counter value minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTPOFF_EL2 : <ul style="list-style-type: none"> EL3 is implemented and SCR_EL3.ECVEn == 0. EL2 is implemented and CNTHCTL_EL2.ECV == 0. 	
0b11	Physical timestamp. The BRBE recorded timestamp is the physical counter value.	

All other values are reserved.

This field is ignored by the PE when EL2 is implemented and [BRBCR_EL2.TS](#) != 0b00.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

MPRED, bit [4]

Mask the recording of mispredicts.

MPRED	Meaning
0b0	Disable the recording of mispredict information.
0b1	Allow the recording of mispredict information.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

CC, bit [3]

Enable the recording of cycle count information.

CC	Meaning
0b0	Disable the recording of cycle count information.
0b1	Allow the recording of cycle count information.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Bit [2]

Reserved, RES0.

E1BRE, bit [1]

EL1 Branch recording enable.

E1BRE	Meaning
0b0	Branch recording prohibited at EL1.
0b1	Branch recording enabled at EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

E0BRE, bit [0]

EL0 Branch recording enable.

E0BRE	Meaning
0b0	Branch recording prohibited at EL0.
0b1	Branch recording enabled at EL0.

This field is ignored by the PE when EL2 is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing BRBCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E0];
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL2;
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL1;

```

MRS <Xt>, BRBCR_EL12

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x8E0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3
trap priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = BRBCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL1;
    else
        UNDEFINED;

```

MSR BRBCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E0] = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        BRBCR_EL2 = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBCR_EL1 = X[t, 64];

```

MSR BRBCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x8E0] = X[t, 64];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            UNDEFINED;
        elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3
trap priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            UNDEFINED;
        elseif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elseif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            BRBCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        BRBCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ERTN	Meaning
0b0	Disable the recording Branch records for exception return instructions from EL2.
0b1	Enable the recording Branch records for exception return instructions from EL2.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Bits [21:109]

Reserved, RES0.

FZPSS, bit [9]

When FEAT_PMUv3_SS is implemented:

Freeze BRBE on PMU Snapshot.

FZPSS	Meaning
0b0	Branch recording is not affected by this control.
0b1	If BRBCR_EL1.FZPSS is 1, then a BRBE freeze event occurs when a PMU snapshot occurs.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FZP, bit [8]

When FEAT_PMUv3 is implemented:

Freeze BRBE on PMU overflow.

FZP	Meaning
0b0	Branch recording is not affected by this control.
0b1	A BRBE freeze event occurs when a PMU overflow occurs.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [7]

Reserved, RES0.

TS, bits [6:5]

Timestamp Control.

TS	Meaning	Applies when
0b00	Timestamp controlled by BRBCR_EL1.TS .	
0b01	Virtual timestamp. The BRBE recorded timestamp is the physical counter value, minus the value of CNTVOFF_EL2 .	
0b10	Guest physical timestamp. The BRBE recorded timestamp is the physical counter value minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTPOFF_EL2 : <ul style="list-style-type: none"> EL3 is implemented and SCR_EL3.ECVEn == 0. EL2 is implemented and CNTHCTL_EL2.ECV == 0. 	When FEAT_ECV is implemented
0b11	Physical timestamp. The BRBE recorded timestamp is the physical counter value.	

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

MPRED, bit [4]

Mask the recording of mispredicts.

MPRED	Meaning
0b0	Disable the recording of mispredict information.
0b1	Allow the recording of mispredict information.

If EL2 is not implemented, then the Effective value of this field is 1, other than for a direct read of the register.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

CC, bit [3]

Enable the recording of cycle count information.

CC	Meaning
0b0	Disable the recording of cycle count information.
0b1	Allow the recording of cycle count information.

If EL2 is not implemented, then the Effective value of this field is 1, other than for a direct read of the register.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Bit [2]

Reserved, RES0.

E2BRE, bit [1]

EL2 Branch recording enable.

E2BRE	Meaning
0b0	Branch recording prohibited at EL2.
0b1	Branch recording enabled at EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

E0HBRE, bit [0]

EL0 Branch recording enable.

E0HBRE	Meaning
0b0	Branch recording prohibited at EL0 when HCR_EL2.TGE == 1.
0b1	Branch recording enabled at EL0 when HCR_EL2.TGE == 1.

This field is ignored by the PE when any of the following are true:

- [HCR_EL2.TGE](#) == 0.
- EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing BRBCR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name BRBCR_EL2 or BRBCR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E0];
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL2;
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL1;

```

MRS <Xt>, BRBCR_EL2

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL2;

```

MSR BRBCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E0] = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        BRBCR_EL2 = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBCR_EL1 = X[t, 64];

```

MSR BRBCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBCR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBFCCR_EL1, Branch Record Buffer Function Control Register

The BRBFCCR_EL1 characteristics are:

Purpose

Functional controls for the Branch Record Buffer.

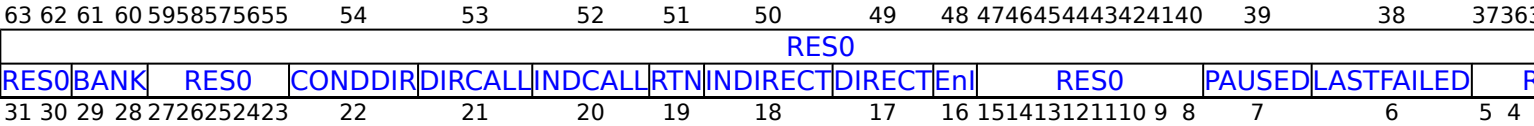
Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBFCCR_EL1 are UNDEFINED.

Attributes

BRBFCCR_EL1 is a 64-bit register.

Field descriptions



Bits [63:30]

Reserved, RES0.

BANK, bits [29:28]

Branch record buffer bank access control.

BANK	Meaning
0b00	Select branch records 0 to 31.
0b01	Select branch records 32 to 63.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:23]

Reserved, RES0.

CONDDIR, bit [22]

Match on conditional direct branch instructions.

CONDDIR	Meaning
0b0	Do not match on conditional direct branch instructions.
0b1	Match on conditional direct branch instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

DIRCALL, bit [21]

Match on direct branch with link instructions.

DIRCALL	Meaning
0b0	Do not match on direct branch with link instructions.
0b1	Match on direct branch with link instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

INDCALL, bit [20]

Match on indirect branch with link instructions.

INDCALL	Meaning
0b0	Do not match on indirect branch with link instructions.
0b1	Match on indirect branch with link instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

RTN, bit [19]

Match on function return instructions.

RTN	Meaning
0b0	Do not match on function return instructions.
0b1	Match on function return instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

INDIRECT, bit [18]

Match on indirect branch instructions.

INDIRECT	Meaning
0b0	Do not match on indirect branch instructions.
0b1	Match on indirect branch instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

DIRECT, bit [17]

Match on unconditional direct branch instructions.

DIRECT	Meaning
0b0	Do not match on unconditional direct branch instructions.
0b1	Match on unconditional direct branch instructions.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

EnI, bit [16]

Include or exclude matches.

EnI	Meaning
0b0	Include records for matches, and exclude records for non-matches.
0b1	Exclude records for matches, and include records for non-matches.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Bits [15:8]

Reserved, RES0.

PAUSED, bit [7]

Branch recording Paused status.

PAUSED	Meaning
0b0	Branch recording is not Paused.
0b1	Branch recording is Paused.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

LASTFAILED, bit [6]**When FEAT_TME is implemented:**

Indicates transaction failure or cancellation.

LASTFAILED	Meaning
0b0	Indicates that no transactions in a non-prohibited region have failed or been canceled since the last Branch record was generated.
0b1	Indicates that at least one transaction in a non-prohibited region has failed or been canceled since the last Branch record was generated.

The reset behavior of this field is:

- On a Cold reset, when FEAT_BRBEv1p1 is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_BRBEv1p1 is not implemented, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:0]

Reserved, RES0.

Accessing BRBFCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBFCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBFCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBFCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBFCR_EL1;

```

MSR BRBFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBFCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBIDR0_EL1, Branch Record Buffer ID0 Register

The BRBIDR0_EL1 characteristics are:

Purpose

Indicates the features of the branch buffer unit.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBIDR0_EL1 are UNDEFINED.

Attributes

BRBIDR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																CC				FORMAT				NUMREC							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

CC, bits [15:12]

Cycle counter support. Defined values are:

CC	Meaning
0b0101	20-bit cycle counter implemented.

All other values are reserved.

FORMAT, bits [11:8]

Data format of records of the Branch record buffer. Defined values are:

FORMAT	Meaning
0b0000	Format 0.

All other values are reserved.

NUMREC, bits [7:0]

Number of records supported. Defined values are:

NUMREC	Meaning
0x08	8 branch records implemented.
0x10	16 branch records implemented.
0x20	32 branch records implemented.
0x40	64 branch records implemented.

All other values are reserved.

Accessing BRBIDR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBIDR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBIDR == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBIDR0_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBIDR0_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = BRBIDR0_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

BRBINFINJ_EL1, Branch Record Buffer Information Injection Register

The BRBINFINJ_EL1 characteristics are:

Purpose

The information of a Branch record for injection.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBINFINJ_EL1 are UNDEFINED.

Attributes

BRBINFINJ_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																	CCU		CC												
RES0														LASTFAILED		T	RES0		TYPE						EL	MPRED	RES0		VALID		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:47]

Reserved, RES0.

CCU, bit [46]

The number of PE clock cycles since the last Branch record entry is UNKNOWN.

CCU	Meaning
0b0	Indicates that the number of PE clock cycles since the last Branch record is indicated by BRBINFINJ_EL1.CC.
0b1	Indicates that the number of PE clock cycles since the last Branch record is UNKNOWN.

The value in this field is only valid when BRBINFINJ_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINFINJ_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

CC, bits [45:32]

The number of PE clock cycles since the last Branch record entry.

The format of this field uses a mantissa and exponent to express the cycle count value, as follows:

- CC bits[7:0] indicate the mantissa M.
- CC bits[13:8] indicate the exponent E.

The cycle count is expressed using the following function:

if IsZero(E) then UInt(M) else UInt('1':M:Zeros(UInt(E)-1))

If required, the cycle count is rounded to a multiple of $2^{(E-1)}$ towards zero before being encoded.

A value of all ones in both the mantissa and exponent indicates the cycle count value exceeded the size of the cycle counter.

The value in this field is only valid when BRBINFINJ_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.CCU == 1
 - BRBINFINJ_EL1.VALID == 0b00
- Otherwise, access to this field is **RW**.

Bits [31:18]

Reserved, RES0.

LASTFAILED, bit [17]

When FEAT_TME is implemented:

Indicates transaction failure or cancellation.

LASTFAILED	Meaning
0b0	Indicates that no transactions in a non-prohibited region have failed or been canceled between the previous Branch record and this Branch record.
0b1	Indicates that at least one transaction in a non-prohibited region has failed or been canceled between the previous Branch record and this Branch record.

The value in this field is only valid when BRBINFINJ_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINFINJ_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

T, bit [16]

When FEAT_TME is implemented:

Transactional state.

T	Meaning
0b0	The branch or exception was not executed in Transactional state.
0b1	The branch or exception was executed in Transactional state.

The value in this field is only valid when BRBINFINJ_EL1.VALID == 0b10 or BRBINFINJ_EL1.VALID == 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.VALID == 0b00
 - BRBINFINJ_EL1.VALID == 0b01
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

Bits [15:14]

Reserved, RES0.

TYPE, bits [13:8]

Branch type.

TYPE	Meaning
0b000000	Unconditional direct branch, excluding Branch with link.
0b000001	Indirect branch, excluding Branch with link, Return from subroutine, and Exception return.
0b000010	Direct Branch with link.
0b000011	Indirect Branch with link.
0b000101	Return from subroutine.
0b000111	Exception return.
0b001000	Conditional direct branch.
0b100001	Debug halt.
0b100010	Call.
0b100011	Trap.
0b100100	SError.
0b100110	Instruction debug.
0b100111	Data debug.
0b101010	Alignment.
0b101011	Inst Fault.
0b101100	Data Fault.
0b101110	IRQ.
0b101111	FIQ.
0b111001	Debug State Exit.

All other values are reserved.

The value in this field is only valid when BRBINFINJ_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINFINJ_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

EL, bits [7:6]

The Exception Level at the target address.

EL	Meaning	Applies when
0b00	EL0.	When FEAT_BRBEv1p1 is implemented
0b01	EL1.	
0b10	EL2.	
0b11	EL3.	

The value in this field is only valid when BRBINFINJ_EL1.VALID == 0b11 or BRBINFINJ_EL1.VALID == 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.VALID == 0b00
 - BRBINFINJ_EL1.VALID == 0b10
- Otherwise, access to this field is **RW**.

MPRED, bit [5]

Branch mispredict.

MPRED	Meaning
0b0	Branch was correctly predicted or the result of the prediction was not captured.
0b1	Branch was incorrectly predicted.

The value in this field is only valid when BRBINFINJ_EL1.VALID == 0b11 or BRBINFINJ_EL1.VALID == 0b10.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.VALID == 0b00
 - BRBINFINJ_EL1.VALID == 0b01
 - BRBINFINJ_EL1.TYPE[5] == 1
- Otherwise, access to this field is **RW**.

Bits [4:2]

Reserved, RES0.

VALID, bits [1:0]

The Branch record is valid.

VALID	Meaning
0b00	This Branch record is not valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBTGTINJ_EL1.ADDRESS. • BRBSRCINJ_EL1.ADDRESS. • BRBINFINJ_EL1.MPRED. • BRBINFINJ_EL1.LASTFAILED. • BRBINFINJ_EL1.T. • BRBINFINJ_EL1.EL. • BRBINFINJ_EL1.TYPE. • BRBINFINJ_EL1.CC. • BRBINFINJ_EL1.CCU.
0b01	This Branch record is valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBSRCINJ_EL1.ADDRESS. • BRBINFINJ_EL1.T. • BRBINFINJ_EL1.MPRED.
0b10	This Branch record is valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBTGTINJ_EL1.ADDRESS. • BRBINFINJ_EL1.EL.
0b11	This Branch record is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing BRBINFINJ_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBINFINJ_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBINFINJ_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBINFINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBINFINJ_EL1;

```

MSR BRBINFINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBINFINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBINFINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBINFINJ_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBINF<n>_EL1, Branch Record Buffer Information Register <n>, n = 0 - 31

The BRBINF<n>_EL1 characteristics are:

Purpose

The information for Branch record n + ([BRBFCCR_EL1](#).BANK × 32).

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBINF<n>_EL1 are UNDEFINED.

Attributes

BRBINF<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																	CCU		CC												
RES0														LASTFAILED		T	RES0		TYPE						EL	MPRED	RES0		VALID		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:47]

Reserved, RES0.

CCU, bit [46]

The number of PE clock cycles since the last Branch record entry is UNKNOWN.

CCU	Meaning
0b0	Indicates that the number of PE clock cycles since the last Branch record is indicated by BRBINF<n>_EL1.CC.
0b1	Indicates that the number of PE clock cycles since the last Branch record is UNKNOWN.

The value in this field is only valid when BRBINF<n>_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINF<n>_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RO**.

CC, bits [45:32]

The number of PE clock cycles since the last Branch record entry.

The format of this field uses a mantissa and exponent to express the cycle count value, as follows:

- CC bits[7:0] indicate the mantissa M.
- CC bits[13:8] indicate the exponent E.

The cycle count is expressed using the following function:

if IsZero(E) then UInt(M) else UInt('1':M:Zeros(UInt(E)-1))

If required, the cycle count is rounded to a multiple of $2^{(E-1)}$ towards zero before being encoded.

A value of all ones in both the mantissa and exponent indicates the cycle count value exceeded the size of the cycle counter.

The value in this field is only valid when BRBINF<n>_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.CCU == 1
 - BRBINF<n>_EL1.VALID == 0b00
- Otherwise, access to this field is **RO**.

Bits [31:18]

Reserved, RES0.

LASTFAILED, bit [17]

When FEAT_TME is implemented:

Indicates transaction failure or cancellation.

LASTFAILED	Meaning
0b0	Indicates that no transactions in a non-prohibited region have failed or been canceled between the previous Branch record and this Branch record.
0b1	Indicates that at least one transaction in a non-prohibited region has failed or been canceled between the previous Branch record and this Branch record.

The value in this field is only valid when BRBINF<n>_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINF<n>_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RO**.

Otherwise:

Reserved, RES0.

T, bit [16]

When FEAT_TME is implemented:

Transactional state.

T	Meaning
0b0	The branch or exception was not executed in Transactional state.
0b1	The branch or exception was executed in Transactional state.

The value in this field is only valid when BRBINF<n>_EL1.VALID == 0b10 or BRBINF<n>_EL1.VALID == 0b11.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.VALID == 0b00
 - BRBINF<n>_EL1.VALID == 0b01
- Otherwise, access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [15:14]

Reserved, RES0.

TYPE, bits [13:8]

Branch type.

TYPE	Meaning
0b000000	Unconditional direct branch, excluding Branch with link.
0b000001	Indirect branch, excluding Branch with link, Return from subroutine, and Exception return.
0b000010	Direct Branch with link.
0b000011	Indirect Branch with link.
0b000101	Return from subroutine.
0b000111	Exception return.
0b001000	Conditional direct branch.
0b100001	Debug halt.
0b100010	Call.
0b100011	Trap.
0b100100	SError.
0b100110	Instruction debug.
0b100111	Data debug.
0b101010	Alignment.
0b101011	Inst Fault.
0b101100	Data Fault.
0b101110	IRQ.
0b101111	FIQ.
0b111001	Debug State Exit.

All other values are reserved.

The value in this field is only valid when BRBINF<n>_EL1.VALID != 0b00.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When BRBINF<n>_EL1.VALID == 0b00, access to this field is **RES0**.
- Otherwise, access to this field is **RO**.

EL, bits [7:6]

The Exception Level at the target address.

EL	Meaning	Applies when
0b00	EL0.	
0b01	EL1.	
0b10	EL2.	
0b11	EL3.	When FEAT_BRBEv1p1 is implemented

The value in this field is only valid when BRBINF<n>_EL1.VALID == 0b11 or BRBINF<n>_EL1.VALID == 0b01.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.VALID == 0b00
 - BRBINF<n>_EL1.VALID == 0b10
- Otherwise, access to this field is **RO**.

MPRED, bit [5]

Branch mispredict.

MPRED	Meaning
0b0	Branch was correctly predicted or the result of the prediction was not captured.
0b1	Branch was incorrectly predicted.

The value in this field is only valid when BRBINF<n>_EL1.VALID == 0b11 or BRBINF<n>_EL1.VALID == 0b10.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.VALID == 0b00
 - BRBINF<n>_EL1.VALID == 0b01
 - BRBINF<n>_EL1.TYPE[5] == 1
- Otherwise, access to this field is **RO**.

Bits [4:2]

Reserved, RES0.

VALID, bits [1:0]

The Branch record is valid.

VALID	Meaning
0b00	This Branch record is not valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBTGT<n>_EL1.ADDRESS. • BRBSRC<n>_EL1.ADDRESS. • BRBINF<n>_EL1.MPRED. • BRBINF<n>_EL1.LASTFAILED. • BRBINF<n>_EL1.T. • BRBINF<n>_EL1.EL. • BRBINF<n>_EL1.TYPE. • BRBINF<n>_EL1.CC. • BRBINF<n>_EL1.CCU.
0b01	This Branch record is valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBSRC<n>_EL1.ADDRESS. • BRBINF<n>_EL1.T. • BRBINF<n>_EL1.MPRED.
0b10	This Branch record is valid. The values of following fields are not valid: <ul style="list-style-type: none"> • BRBTGT<n>_EL1.ADDRESS. • BRBINF<n>_EL1.EL.
0b11	This Branch record is valid.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing BRBINF<n>_EL1

BRBINF<n>_EL1 reads-as-zero if $n + (\text{BRBFCR_EL1.BANK} \times 32) \geq \text{BRBIDR0_EL1.NUMREC}$.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBINF<m>_EL1 ; Where m = 0-31

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:0b00

```

integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBSRCINJ_EL1, Branch Record Buffer Source Address Injection Register

The BRBSRCINJ_EL1 characteristics are:

Purpose

The source address of a Branch record for injection.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBSRCINJ_EL1 are UNDEFINED.

Attributes

BRBSRCINJ_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDRESS																															
ADDRESS																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Source virtual address of the Branch record.

When a direct write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an UNKNOWN value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by VAMax(). The value in bits [P-1:0] are the value written.

When a direct write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.VALID == 0b00
 - BRBINFINJ_EL1.VALID == 0b01
- Otherwise, access to this field is **RW**.

Accessing BRBSRCINJ_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBSRCINJ_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b10	0b001	0b1001	0b0001	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBSRCINJ_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBSRCINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBSRCINJ_EL1;

```

MSR BRBSRCINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBSRCINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBSRCINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBSRCINJ_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBSRC<n>_EL1, Branch Record Buffer Source Address Register <n>, n = 0 - 31

The BRBSRC<n>_EL1 characteristics are:

Purpose

The source address of Branch record n + ([BRBFCCR_EL1](#).BANK × 32).

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBSRC<n>_EL1 are UNDEFINED.

Attributes

BRBSRC<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDRESS																															
ADDRESS																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Source virtual address of the Branch record.

When an indirect write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an UNKNOWN value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by VAMax(). The value in bits [P-1:0] are the value written.

When an indirect write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.VALID == 0b00
 - BRBINF<n>_EL1.VALID == 0b01
- Otherwise, access to this field is **RO**.

Accessing BRBSRC<n>_EL1

BRBSRC<n>_EL1 is RES0 if n + ([BRBFCCR_EL1](#).BANK × 32) >= [BRBIDR0_EL1](#).NUMREC.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBSRC<m>_EL1 ; Where m = 0-31

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:0b01

```

integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBTGTINJ_EL1, Branch Record Buffer Target Address Injection Register

The BRBTGTINJ_EL1 characteristics are:

Purpose

The target address of a Branch record for injection.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBTGTINJ_EL1 are UNDEFINED.

Attributes

BRBTGTINJ_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDRESS																															
ADDRESS																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Target virtual address of the Branch record.

When a direct write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an UNKNOWN value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by VAMax(). The value in bits [P-1:0] are the value written.

When a direct write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINFINJ_EL1.VALID == 0b00
 - BRBINFINJ_EL1.VALID == 0b10
- Otherwise, access to this field is **RW**.

Accessing BRBTGTINJ_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBTGTINJ_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b10	0b001	0b1001	0b0001	0b010
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTGTINJ_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTGTINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBTGTINJ_EL1;

```

MSR BRBTGTINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTGTINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTGTINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBTGTINJ_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBTGT<n>_EL1, Branch Record Buffer Target Address Register <n>, n = 0 - 31

The BRBTGT<n>_EL1 characteristics are:

Purpose

The target address of Branch record n + ([BRBFCCR_EL1](#).BANK × 32).

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBTGT<n>_EL1 are UNDEFINED.

Attributes

BRBTGT<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDRESS																															
ADDRESS																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Target virtual address of the Branch record.

When an indirect write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an UNKNOWN value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by VAMax(). The value in bits [P-1:0] are the value written.

When an indirect write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES0** if any of the following are true:
 - BRBINF<n>_EL1.VALID == 0b00
 - BRBINF<n>_EL1.VALID == 0b10
- Otherwise, access to this field is **RO**.

Accessing BRBTGT<n>_EL1

BRBTGT<n>_EL1 is RES0 if n + ([BRBFCCR_EL1](#).BANK × 32) >= [BRBIDRO_EL1](#).NUMREC.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBTGT<m>_EL1 ; Where m = 0-31

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:0b10

```

integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBTS_EL1, Branch Record Buffer Timestamp Register

The BRBTS_EL1 characteristics are:

Purpose

Captures the Timestamp value on a BRBE freeze event.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBTS_EL1 are UNDEFINED.

Attributes

BRBTS_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																TS															
																TS															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TS, bits [63:0]

Timestamp value at the time of a BRBE freeze event.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing BRBTS_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, BRBTS_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTS_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTS_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = BRBTS_EL1;

```

MSR BRBTS_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTS_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTS_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBTS_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CCSIDR_EL1, Current Cache Size ID Register

The CCSIDR_EL1 characteristics are:

Purpose

Provides information about the architecture of the currently selected cache.

Configuration

AArch64 System register CCSIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CCSIDR\[31:0\]](#).

AArch64 System register CCSIDR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [CCSIDR2\[31:0\]](#).

The implementation includes one CCSIDR_EL1 for each cache that it can access. [CSSELR_EL1](#) selects which Cache Size ID Register is accessible.

Attributes

CCSIDR_EL1 is a 64-bit register.

Field descriptions

When FEAT_CCIDX is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
RES0								NumSets																										
RES0								Associativity																									LineSize	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

Note

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a design. You cannot make any inference about the actual sizes of caches based on these parameters.

Bits [63:56]

Reserved, RES0.

NumSets, bits [55:32]

(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.

Bits [31:24]

Reserved, RES0.

Associativity, bits [23:3]

(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

LineSize, bits [2:0]

($\log_2(\text{Number of bytes in cache line})$) - 4. For example:

- For a line length of 16 bytes: $\log_2(16) = 4$, LineSize entry = 0. This is the minimum line length.
- For a line length of 32 bytes: $\log_2(32) = 5$, LineSize entry = 1.

Note

The C++ 17 specification has two defined parameters relating to the granularity of memory that does not interfere. For generic software and tools, Arm will set the hardware_destructive_interference_size parameter to 256 bytes and the hardware_constructive_interference_size parameter to 64 bytes.

When FEAT_MTE2 is implemented **and enabled**, where a cache only holds Allocation tags, this field is RES0.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
UNKNOWN				NumSets																Associativity								LineSize			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a design. You cannot make any inference about the actual sizes of caches based on these parameters.

Bits [63:32]

Reserved, RES0.

Bits [31:28]

Reserved, UNKNOWN.

NumSets, bits [27:13]

(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.

Associativity, bits [12:3]

(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

LineSize, bits [2:0]

($\log_2(\text{Number of bytes in cache line})$) - 4. For example:

- For a line length of 16 bytes: $\log_2(16) = 4$, LineSize entry = 0. This is the minimum line length.
- For a line length of 32 bytes: $\log_2(32) = 5$, LineSize entry = 1.

When FEAT_MTE2 is implemented, where a cache only holds Allocation tags, this field is RES0.

Note

The C++ 17 specification has two defined parameters relating to the granularity of memory that does not interfere. For generic software and tools, Arm will set the hardware_destructive_interference_size parameter to 256 bytes and the hardware_constructive_interference_size parameter to 64 bytes.

Accessing CCSIDR_EL1

If [CSELR_EL1](#).{TnD, Level, InD} is programmed to a cache level that is not implemented, then on a read of the CCSIDR_EL1 the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:

- The CCSIDR_EL1 read is treated as NOP.
- The CCSIDR_EL1 read is UNDEFINED.
- The CCSIDR_EL1 read returns an UNKNOWN value.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CCSIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.CCSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CCSIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CCSIDR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = CCSIDR_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CFP RCTX, Control Flow Prediction Restriction by Context

The CFP RCTX characteristics are:

Purpose

Control Flow Prediction Restriction by Context applies to all Control Flow Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Control flow predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to CFP RCTX are UNDEFINED.

Attributes

CFP RCTX is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0															GVMID	VMID															
RES0				NSE	NS	EL		RES0								GASID	ASID														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]

When FEAT_RME is implemented:

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see CFP_RCTX.NS.

Otherwise:

Reserved, RES0.

NS, bit [26]

When FEAT_RME is implemented:

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.
- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.

- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

This instruction is treated as a NOP when executed at EL3 and either:

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with CFP_RCTX.{NSE, NS} == {1, 0}.

- CFP_RCTX.{NSE, NS} selects a reserved value.
- CFP_RCTX.{NSE, NS} == {1, 0} and CFP_RCTX.EL has a value other than 0b11.

Otherwise:

Security State. Defined values are:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Executing the CFP RCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

CFP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b100

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CFPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnRCTX == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.CFPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CLIDR_EL1, Cache Level ID Register

The CLIDR_EL1 characteristics are:

Purpose

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configuration

AArch64 System register CLIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CLIDR\[31:0\]](#).

Attributes

CLIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																Ttype7	Ttype6	Ttype5	Ttype4	Ttype3	Ttype2	Ttype1	ICB								
ICB		LoUU		LoC		LoUIS		Ctype7		Ctype6		Ctype5		Ctype4		Ctype3		Ctype2		Ctype1											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:47]

Reserved, RES0.

Ttype<n>, bits [2(n-1)+34:2(n-1)+33], for n = 7 to 1 When FEAT_MTE2 is implemented:

Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.

Ttype<n>	Meaning
0b00	No Tag Cache.
0b01	Separate Allocation Tag Cache.
0b10	Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines.
0b11	Unified Allocation Tag and Data cache, Allocation Tags and Data in separate lines.

Otherwise:

Reserved, RES0.

ICB, bits [32:30]

Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.

ICB	Meaning
0b000	Not disclosed by this mechanism.
0b001	L1 cache is the highest Inner Cacheable level.
0b010	L2 cache is the highest Inner Cacheable level.
0b011	L3 cache is the highest Inner Cacheable level.
0b100	L4 cache is the highest Inner Cacheable level.
0b101	L5 cache is the highest Inner Cacheable level.
0b110	L6 cache is the highest Inner Cacheable level.
0b111	L7 cache is the highest Inner Cacheable level.

LoUU, bits [29:27]

Level of Unification Uniprocessor for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

Note

This field does not describe the requirements for instruction cache invalidation. See [CTR_EL0.DIC](#).

Note

When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches.

LoC, bits [26:24]

Level of Coherence for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

LoUIS, bits [23:21]

Level of Unification Inner Shareable for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

Note

This field does not describe the requirements for instruction cache invalidation. See [CTR_EL0.DIC](#).

Note

When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches.

Ctype<n>, bits [3(n-1)+2:3(n-1)], for n = 7 to 1

Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are:

Ctype<n>	Meaning
0b000	No cache.
0b001	Instruction cache only.
0b010	Data cache only.
0b011	Separate instruction and data caches.
0b100	Unified cache.

All other values are reserved.

If software reads the Cache Type fields from Ctype1 upwards, once it has seen a value of 000, no caches that can be managed using the architected cache maintenance instructions that operate by set/way exist at further-out levels of the hierarchy. So, for example, if Ctype3 is the first Cache Type field with a value of 000, the values of Ctype4 to Ctype7 must be ignored.

Accessing CLIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CLIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b001

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
        && HFGTR_EL2.CLIDR_EL1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CLIDR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = CLIDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = CLIDR_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHCTL_EL2, Counter-timer Hypervisor Control register

The CNTHCTL_EL2 characteristics are:

Purpose

Controls the generation of an event stream from the physical counter, and access from EL1 to the physical counter and the EL1 physical timer.

Configuration

AArch64 System register CNTHCTL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHCTL\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

CNTHCTL_EL2 is a 64-bit register.

Field descriptions

When FEAT_VHE is implemented and HCR_EL2.E2H == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	
																		RES0				
RES0												CNTPMASK	CNTVMASK	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVT	ECV	EL1PTEN	EL1PCTE	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	

Bits [63:20]

Reserved, RES0.

CNTPMASK, bit [19]

When FEAT_RME is implemented:

CNTPMASK	Meaning
0b0	This control has no affect on CNTP_CTL_EL0.IMASK .
0b1	CNTP_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CNTVMASK, bit [18]**When FEAT_RME is implemented:**

CNTVMASK	Meaning
0b0	This control has no affect on CNTV_CTL_EL0 .IMASK.
0b1	CNTV_CTL_EL0 .IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENTIS, bit [17]**When FEAT_ECV is implemented:**

Controls the scale of the generation of the event stream.

EVENTIS	Meaning
0b0	The CNTHCTL_EL2.EVENTI field applies to CNTPCT_EL0 [15:0].
0b1	The CNTHCTL_EL2.EVENTI field applies to CNTPCT_EL0 [23:8].

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1NVVCT, bit [16]**When FEAT_ECV is implemented:**

Traps EL1 accesses to the specified EL1 virtual timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

EL1NVVCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) HCR_EL2.NV2==0 HCR_EL2.NV1==1 HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If ((HCR_EL2.E2H==0 HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1), then EL1 accesses to CNTV_CTL_EL02 and CNTV_CVAL_EL02 are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1NVPCT, bit [15]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 physical timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

EL1NVPCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H==1 && HCR_EL2.TGE==1) HCR_EL2.NV2==0 HCR_EL2.NV1==1 HCR_EL2.NV==0), this control does not cause any instructions to be trapped. If (HCR_EL2.E2H==0 HCR_EL2.TGE==0) && HCR_EL2.NV2==1 && HCR_EL2.NV1==0 && HCR_EL2.NV==1 , then EL1 accesses to CNTP_CTL_EL02 and CNTP_CVAL_EL02, are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1TVCT, bit [14]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual counter registers to EL2, when EL2 is enabled for the current Security state.

EL1TVCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	<p>If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.</p> <p>If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then:</p> <ul style="list-style-type: none"> In AArch64 state, traps EL0 and EL1 accesses to CNTVCT_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN. In AArch32 state, traps EL0 and EL1 accesses to CNTVCT to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN or CNTKCTL.PL0VCTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1TVT, bit [13]

When FEAT_ECV is implemented:

Traps EL0 and EL1 accesses to the EL1 virtual timer registers to EL2, when EL2 is enabled for the current Security state.

EL1TVT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	<p>If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.</p> <p>If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then:</p> <ul style="list-style-type: none"> In AArch64 state, traps EL0 and EL1 accesses to CNTV_CTL_EL0, CNTV_CVAL_EL0, and CNTV_TVAL_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN. In AArch32 state, traps EL0 and EL1 accesses to CNTV_CTL, CNTV_CVAL, and CNTV_TVAL to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN or CNTKCTL.PL0VTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ECV, bit [12]

When FEAT_ECV is implemented:

Enables the Enhanced Counter Virtualization functionality registers.

ECV	Meaning
0b0	Enhanced Counter Virtualization functionality is disabled.
0b1	When HCR_EL2 .{E2H, TGE} == {1, 1} or SCR_EL3 .{NS, EEL2} == {0, 0}, then Enhanced Counter Virtualization functionality is disabled. When SCR_EL3 .NS or SCR_EL3 .EEL2 are 1, and HCR_EL2 .E2H or HCR_EL2 .TGE are 0, then Enhanced Counter Virtualization functionality is enabled when EL2 is enabled for the current Security state. This means that: <ul style="list-style-type: none"> An MRS to CNTPCT_EL0 from either EL0 or EL1 that is not trapped will return the value (PCount<63:0> - CNTPOFF_EL2<63:0>). The EL1 physical timer interrupt is triggered when ((PCount<63:0> - CNTPOFF_EL2<63:0>) - PCVal<63:0>) is greater than or equal to 0. PCount<63:0> is the physical count returned when CNTPCT_EL0 is read from EL2 or EL3. PCVal<63:0> is the EL1 physical timer compare value for this timer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1PTEN, bit [11]

When [HCR_EL2](#).TGE is 0, traps EL0 and EL1 accesses to the E1 physical timer registers to EL2 when EL2 is enabled in the current Security state.

EL1PTEN	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTP_CTL_EL0 , CNTP_CVAL_EL0 , and CNTP_TVAL_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1 .ELOPTEN. From AArch32 state: EL0 and EL1 accesses to the CNTP_CTL , CNTP_CVAL , and CNTP_TVAL are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1 .ELOPTEN or CNTKCTL .PLOPTEN.
0b1	This control does not cause any instructions to be trapped.

When [HCR_EL2](#).TGE is 1, this control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCTEN, bit [10]

When [HCR_EL2](#).TGE is 0, traps EL0 and EL1 accesses to the EL1 physical counter register to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to [CNTPCT_EL0](#) are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRRC or MCRR accesses to [CNTPCT](#) are trapped to EL2, reported using EC syndrome value 0x04.

EL1PCTEN	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTPCT_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN . From AArch32 state: EL0 and EL1 accesses to the CNTPCT are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN or CNTKCTL.PL0PCTEN .
0b1	This control does not cause any instructions to be trapped.

When [HCR_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ELOPTEN, bit [9]

When [HCR_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR_EL2.TGE](#) is 1, traps EL0 accesses to the physical timer registers to EL2.

ELOPTEN	Meaning
0b0	EL0 using AArch64: EL0 accesses to the CNTP_CTL_EL0 , CNTP_CVAL_EL0 , and CNTP_TVAL_EL0 registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTP_CTL , CNTP_CVAL and CNTP_TVAL registers are trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ELOVTEN, bit [8]

When [HCR_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR_EL2.TGE](#) is 1, traps EL0 accesses to the virtual timer registers to EL2.

ELOVTEN	Meaning
0b0	EL0 using AArch64: EL0 accesses to the CNTV_CTL_EL0 , CNTV_CVAL_EL0 , and CNTV_TVAL_EL0 registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTV_CTL , CNTV_CVAL , and CNTV_TVAL registers are trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTI, bits [7:4]

Selects which bit of [CNTPCT_EL0](#), as seen from EL2, is the trigger for the event stream generated from that counter when that stream is enabled.

If FEAT_ECV is implemented, and CNTHCTL_EL2.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of [CNTPCT_EL0](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of [CNTPCT_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTDIR, bit [3]

Controls which transition of the [CNTPCT_EL0](#) trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is enabled.

EVNTDIR	Meaning
0b0	A 0 to 1 transition of the trigger bit triggers an event.
0b1	A 1 to 0 transition of the trigger bit triggers an event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTEN, bit [2]

Enables the generation of an event stream from [CNTPCT_EL0](#) as seen from EL2.

EVNTEN	Meaning
0b0	Disables the event stream.
0b1	Enables the event stream.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ELOVCTEN, bit [1]

When [HCR_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR_EL2.TGE](#) is 1, traps EL0 accesses to the frequency register and virtual counter register to EL2.

ELOVCTEN	Meaning
0b0	EL0 using AArch64: EL0 accesses to the CNTVCT_EL0 are trapped to EL2. EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0PCTEN is also 0. EL0 using AArch32: EL0 accesses to the CNTVCT are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTFRQ register are trapped to EL2, if CNTHCTL.EL0PCTEN is also 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL0PCTEN, bit [0]

When [HCR_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR_EL2.TGE](#) is 1, traps EL0 accesses to the frequency register and physical counter register to EL2.

EL0PCTEN	Meaning
0b0	EL0 using AArch64: EL0 accesses to the CNTPCT_EL0 are trapped to EL2. EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0. EL0 using AArch32: EL0 accesses to the CNTPCT are trapped to EL2. EL0 using AArch32: EL0 accesses to the CNTFRQ and register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

636261605958575655545352	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20
RES0																																
RES0		CNTPMASK	CNTVMASK	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVTECV	RES0	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVTECV	RES0	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVTECV	RES0	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVTECV	RES0	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVTECV
313029282726252423222120	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20

The following format field descriptions apply in all Armv8.0 implementations.

It also explains the behavior when EL3 is implemented and EL2 is not implemented.

Bits [63:20]

Reserved, RES0.

CNTPMASK, bit [19]

When FEAT_RME is implemented:

CNTPMASK	Meaning
0b0	This control has no affect on CNTP_CTL_EL0.IMASK .
0b1	CNTP_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CNTVMASK, bit [18]

When FEAT_RME is implemented:

CNTVMASK	Meaning
0b0	This control has no affect on CNTV_CTL_EL0.IMASK .
0b1	CNTV_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.

This bit is RES0 in Non-secure and Secure state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVNTIS, bit [17]

When FEAT_ECV is implemented:

Controls the scale of the generation of the event stream.

EVNTIS	Meaning
0b0	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_ELO [15:0].
0b1	The CNTHCTL_EL2.EVNTI field applies to CNTPCT_ELO [23:8].

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1NVVCT, bit [16]

When FEAT_ECV is implemented:

Traps EL1 accesses to the specified EL1 virtual timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

EL1NVVCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H ==1 && HCR_EL2.TGE ==1) HCR_EL2.NV2 ==0 HCR_EL2.NV1 ==1 HCR_EL2.NV ==0), this control does not cause any instructions to be trapped. If ((HCR_EL2.E2H ==0 HCR_EL2.TGE ==0) && HCR_EL2.NV2 ==1 && HCR_EL2.NV1 ==0 && HCR_EL2.NV ==1), then EL1 accesses to CNTV_CTL_EL02 and CNTV_CVAL_EL02 are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1NVPCT, bit [15]**When FEAT_ECV is implemented:**

Traps EL1 accesses to the specified EL1 physical timer registers using the EL02 descriptors to EL2, when EL2 is enabled for the current Security state.

EL1NVPCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If ((HCR_EL2.E2H ==1 && HCR_EL2.TGE ==1) HCR_EL2.NV2 ==0 HCR_EL2.NV1 ==1 HCR_EL2.NV ==0), this control does not cause any instructions to be trapped. If (HCR_EL2.E2H ==0 HCR_EL2.TGE ==0) && HCR_EL2.NV2 ==1 && HCR_EL2.NV1 ==0 && HCR_EL2.NV ==1, then EL1 accesses to CNTP_CTL_EL02 and CNTP_CVAL_EL02, are trapped to EL2.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1TVCT, bit [14]**When FEAT_ECV is implemented:**

Traps EL0 and EL1 accesses to the EL1 virtual counter registers to EL2, when EL2 is enabled for the current Security state.

EL1TVCT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	If HCR_EL2 .{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped. If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then: In AArch64 state, traps EL0 and EL1 accesses to CNTVCT_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN . In AArch32 state, traps EL0 and EL1 accesses to CNTVCT to EL2, unless they are trapped by CNTKCTL_EL1.EL0VCTEN or CNTKCTL.PL0VCTEN .

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EL1TVT, bit [13]**When FEAT_ECV is implemented:**

Traps EL0 and EL1 accesses to the EL1 virtual timer registers to EL2, when EL2 is enabled for the current Security state.

EL1TVT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	<p>If HCR_EL2.{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.</p> <p>If HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, then:</p> <ul style="list-style-type: none"> In AArch64 state, traps EL0 and EL1 accesses to CNTV_CTL_EL0, CNTV_CVAL_EL0, and CNTV_TVAL_EL0 to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN. In AArch32 state, traps EL0 and EL1 accesses to CNTV_CTL, CNTV_CVAL, and CNTV_TVAL to EL2, unless they are trapped by CNTKCTL_EL1.EL0VTEN or CNTKCTL.PL0VTEN.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 0 other than for the purpose of a direct read.

This control applies regardless of the value of the CNTHCTL_EL2.ECV bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ECV, bit [12]**When FEAT_ECV is implemented:**

Enables the Enhanced Counter Virtualization functionality registers.

ECV	Meaning
0b0	Enhanced Counter Virtualization functionality is disabled.
0b1	<p>When HCR_EL2.{E2H, TGE} == {1, 1} or SCR_EL3.{NS, EEL2} == {0, 0}, then Enhanced Counter Virtualization functionality is disabled.</p> <p>When SCR_EL3.NS or SCR_EL3.EEL2 are 1, and HCR_EL2.E2H or HCR_EL2.TGE are 0, then Enhanced Counter Virtualization functionality is enabled when EL2 is enabled for the current Security state. This means that:</p> <ul style="list-style-type: none"> An MRS to CNTPCT_EL0 from either EL0 or EL1 that is not trapped will return the value (PCount<63:0> - CNTPOFF_EL2<63:0>). The EL1 physical timer interrupt is triggered when ((PCount<63:0> - CNTPOFF_EL2<63:0>) - PCVal<63:0>) is greater than or equal to 0. PCount is the physical count returned when CNTPCT_EL0 is read from EL2 or EL3. PCVal<63:0> is the EL1 physical timer compare value for this timer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [11:8]

Reserved, RES0.

EVNTI, bits [7:4]

Selects which bit of [CNTPCT_EL0](#), as seen from EL2, is the trigger for the event stream generated from that counter when that stream is enabled.

If FEAT_ECV is implemented, and CNTHCTL_EL2.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of [CNTPCT_EL0](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of [CNTPCT_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTDIR, bit [3]

Controls which transition of the [CNTPCT_EL0](#) trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is enabled.

EVNTDIR	Meaning
0b0	A 0 to 1 transition of the trigger bit triggers an event.
0b1	A 1 to 0 transition of the trigger bit triggers an event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EVNTEN, bit [2]

Enables the generation of an event stream from [CNTPCT_EL0](#) as seen from EL2.

EVNTEN	Meaning
0b0	Disables the event stream.
0b1	Enables the event stream.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCEN, bit [1]

Traps EL0 and EL1 accesses to the EL1 physical timer registers to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to [CNTP_CTL_EL0](#), [CNTP_CVAL_EL0](#), [CNTP_TVAL_EL0](#) are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2 reported using EC syndrome value 0x3 and MRRC and MCRR accesses are trapped to EL2, reported using EC syndrome value 0x04:
 - [CNTP_CTL](#), [CNTP_CVAL](#), [CNTP_TVAL](#).

EL1PCEN	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTP_CTL_EL0 , CNTP_CVAL_EL0 , and CNTP_TVAL_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN . From AArch32 state: EL0 and EL1 accesses to the CNTP_CTL , CNTP_CVAL , and CNTP_TVAL are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PTEN or CNTKCTL.PL0PTEN .
0b1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EL1PCTEN, bit [0]

Traps EL0 and EL1 accesses to the EL1 physical counter register to EL2 when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to [CNTPCT_EL0](#) are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRRC or MCRR accesses to [CNTPCT](#) are trapped to EL2, reported using EC syndrome value 0x04.

EL1PCTEN	Meaning
0b0	From AArch64 state: EL0 and EL1 accesses to the CNTPCT_EL0 are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN . From AArch32 state: EL0 and EL1 accesses to the CNTPCT are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by CNTKCTL_EL1.EL0PCTEN or CNTKCTL.PL0PCTEN .
0b1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CNTHCTL_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic [CNTHCTL_EL2](#) or [CNTKCTL_EL1](#) are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CNTHCTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0001	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CNTHCTL_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CNTHCTL_EL2;

```

MSR CNTHCTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHCTL_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CNTHCTL_EL2 = X[t, 64];

```

MRS <Xt>, CNTKCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1110	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = CNTKCTL_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = CNTHCTL_EL2;
    else
        X[t, 64] = CNTKCTL_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CNTKCTL_EL1;

```

MSR CNTKCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1110	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    CNTKCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CNTHCTL_EL2 = X[t, 64];
    else
        CNTKCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CNTKCTL_EL1 = X[t, 64];

```

3095/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

CNTPS_CTL_EL1, Counter-timer Physical Secure Timer Control register

The CNTPS_CTL_EL1 characteristics are:

Purpose

Control register for the secure physical timer, usually accessible at EL3 but configurably accessible at EL1 in Secure state.

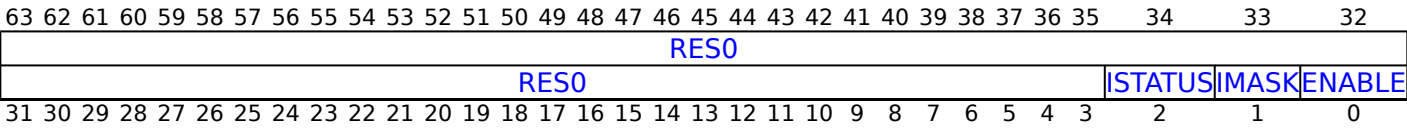
Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to CNTPS_CTL_EL1 are undefined.

Attributes

CNTPS_CTL_EL1 is a 64-bit register.

Field descriptions



Bits [63:3]

Reserved, RES0.

ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0b0	Timer condition is not met.
0b1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0b0	Timer interrupt is not masked by the IMASK bit.
0b1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0b0	Timer disabled.
0b1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTPS_TVAL_EL1](#) continues to count down.

Note

Disabling the output signal might be a power-saving option.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CNTPS_CTL_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CNTPS_CTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = CNTPS_CTL_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = CNTPS_CTL_EL1;

```

MSR CNTPS_CTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            CNTPS_CTL_EL1 = X[t, 64];
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        CNTPS_CTL_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTPS_CVAL_EL1, Counter-timer Physical Secure Timer CompareValue register

The CNTPS_CVAL_EL1 characteristics are:

Purpose

Holds the compare value for the secure physical timer, usually accessible at EL3 but configurably accessible at EL1 in Secure state.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to CNTPS_CVAL_EL1 are undefined.

Attributes

CNTPS_CVAL_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
CompareValue																															
CompareValue																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CompareValue, bits [63:0]

Holds the secure physical timer CompareValue.

When CNTPS_CTL_EL1.ENABLE is 1, the timer condition is met when (CNTPCT_EL0 - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:

- CNTPS_CTL_EL1.ISTATUS is set to 1.
- If CNTPS_CTL_EL1.IMASK is 0, an interrupt is generated.

When CNTPS_CTL_EL1.ENABLE is 0, the timer condition is not met, but CNTPCT_EL0 continues to count.

If the Generic counter is implemented at a size less than 64 bits, then this field is permitted to be implemented at the same width as the counter, and the upper bits are RES0.

The value of this field is treated as zero-extended in all counter calculations.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CNTPS_CVAL_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CNTPS_CVAL_EL1

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = CNTPS_CVAL_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = CNTPS_CVAL_EL1;

```

MSR CNTPS_CVAL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            CNTPS_CVAL_EL1 = X[t, 64];
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        CNTPS_CVAL_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The CNTPS_TVAL_EL1 characteristics are:

Purpose

Holds the timer value for the secure physical timer, usually accessible at EL3 but configurably accessible at EL1 in Secure state.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to CNTPS TVAL EL1 are **UNDEFINED**.

Attributes

CNTPS_TVAL_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
TimerValue																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

TimerValue, bits [31:0]

The TimerValue view of the secure physical timer.

On a read of this register:

- If `CNTPS_CTL_EL1.ENABLE` is 0, the value returned is `UNKNOWN`.
- If `CNTPS_CTL_EL1.ENABLE` is 1, the value returned is `(CNTPS_CVAL_EL1 - CNTPCT_EL0)`.

On a write of this register, [CNTPS_CVAL_EL1](#) is set to ([CNTPCT_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When `CNTPS_CTL_EL1.ENABLE` is 1, the timer condition is met when `(CNTPCT_ELO - CNTPS_CVAL_EL1)` is greater than or equal to zero. This means that `TimerValue` acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTPS_CTL_EL1](#).ISTATUS is set to 1.
- If [CNTPS_CTL_EL1](#).IMASK is 0, an interrupt is generated.

When `CNTPS_CTL_EL1.ENABLE` is 0, the timer condition is not met, but `CNTPCT_EL0` continues to count, so the `TimerValue` view appears to continue to count down.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CNTPS_TVAL_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CNTPS_TVAL_EL1

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
        CNTHCTL_EL2.ECV == '1' then
            if CNTPS_CTL_EL1.ENABLE == '0' then
                X[t, 64] = bits(64) UNKNOWN;
            else
                X[t, 64] = CNTPS_CVAL_EL1 - (PhysicalCountInt() - CNTPOFF_EL2);
            else
                if CNTPS_CTL_EL1.ENABLE == '0' then
                    X[t, 64] = bits(64) UNKNOWN;
                else
                    X[t, 64] = CNTPS_CVAL_EL1 - PhysicalCountInt();
            else
                UNDEFINED;
        elsif PSTATE.EL == EL2 then
            UNDEFINED;
        elsif PSTATE.EL == EL3 then
            if CNTPS_CTL_EL1.ENABLE == '0' then
                X[t, 64] = bits(64) UNKNOWN;
            else
                X[t, 64] = CNTPS_CVAL_EL1 - PhysicalCountInt();

```

MSR CNTPS_TVAL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b111	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && SCR_EL3.NS == '0' then
        if SCR_EL3.EEL2 == '1' then
            UNDEFINED;
        elsif SCR_EL3.ST == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
        CNTHCTL_EL2.ECV == '1' then
            CNTPS_CVAL_EL1 = (SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt()) - CNTPOFF_EL2;
        else
            CNTPS_CVAL_EL1 = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        UNDEFINED;
    elsif PSTATE.EL == EL3 then
        CNTPS_CVAL_EL1 = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

CONTEXTIDR_EL1, Context ID Register (EL1)

The CONTEXTIDR_EL1 characteristics are:

Purpose

Identifies the current Process Identifier.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

Configuration

AArch64 System register CONTEXTIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CONTEXTIDR\[31:0\]](#).

Attributes

CONTEXTIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																PROCID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

PROCID, bits [31:0]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

Note

In AArch32 state, when [TTBCR](#).EAE is set to 0, [CONTEXTIDR](#).ASID holds the ASID.

In AArch64 state, CONTEXTIDR_EL1 is independent of the ASID, and for the EL1&0 translation regime either [TTBR0_EL1](#) or [TTBR1_EL1](#) holds the ASID.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CONTEXTIDR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic [CONTEXTIDR_EL1](#) or [CONTEXTIDR_EL12](#) are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CONTEXTIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.CONTEXTIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x108];
    else
        X[t, 64] = CONTEXTIDR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = CONTEXTIDR_EL2;
    else
        X[t, 64] = CONTEXTIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CONTEXTIDR_EL1;

```

MSR CONTEXTIDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.CONTEXTIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x108] = X[t, 64];
    else
        CONTEXTIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CONTEXTIDR_EL2 = X[t, 64];
    else
        CONTEXTIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CONTEXTIDR_EL1 = X[t, 64];

```

MRS <Xt>, CONTEXTIDR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x108];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = CONTEXTIDR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = CONTEXTIDR_EL1;
    else
        UNDEFINED;

```

MSR CONTEXTIDR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x108] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CONTEXTIDR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        CONTEXTIDR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CONTEXTIDR_EL2, Context ID Register (EL2)

The CONTEXTIDR_EL2 characteristics are:

Purpose

Identifies the current Process Identifier for EL2.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

Configuration

This register is present only when FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented. Otherwise, direct accesses to CONTEXTIDR_EL2 are UNDEFINED.

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

CONTEXTIDR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																PROCID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

PROCID, bits [31:0]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CONTEXTIDR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CONTEXTIDR_EL2 or CONTEXTIDR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CONTEXTIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CONTEXTIDR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CONTEXTIDR_EL2;

```

MSR CONTEXTIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    CONTEXTIDR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CONTEXTIDR_EL2 = X[t, 64];

```

MRS <Xt>, CONTEXTIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.CONTEXTIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x108];
    else
        X[t, 64] = CONTEXTIDR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = CONTEXTIDR_EL2;
    else
        X[t, 64] = CONTEXTIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CONTEXTIDR_EL1;

```

MSR CONTEXTIDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.CONTEXTIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x108] = X[t, 64];
    else
        CONTEXTIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CONTEXTIDR_EL2 = X[t, 64];
    else
        CONTEXTIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CONTEXTIDR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

COSP RCTX, Clear Other Speculative Restriction by Context

The COSP RCTX characteristics are:

Purpose

Clear Other Speculative Prediction Restriction by Context applies to all prediction resources not managed by another of the speculation restriction System instructions.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control any predictions occurring after the instruction is complete and synchronized.

This instruction applies to all speculative access except:

- Cache Prefetch predictions.
- Data Value predictions.
- Control Flow predictions.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when FEAT_SPECRES2 is implemented. Otherwise, direct accesses to COSP RCTX are UNDEFINED.

Attributes

COSP RCTX is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0															GVMID	VMID																
RES0				NSE	NS	EL		RES0							GASID	ASID																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 and EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]**When FEAT_RME is implemented:**

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see [COSP_RCTX.NS](#).

Otherwise:

Reserved, RES0.

NS, bit [26]**When FEAT_RME is implemented:**

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.
- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.
- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

This instruction is treated as a NOP when executed at EL3 and either:

- COSP_RCTX.{NSE, NS} selects a reserved value.
- COSP_RCTX.{NSE, NS} == {1, 0} and COSP_RCTX.EL has a value other than 0b11.

Otherwise:

Security State. Defined values are:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Executing COSP RCTX

Accesses to this instruction use the following encodings in the System instruction encoding space:

COSP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b110

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.COSPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.EnRCTX == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.COSPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_Other);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPACR_EL1, Architectural Feature Access Control Register

The CPACR_EL1 characteristics are:

Purpose

Controls access to trace, SME, Streaming SVE, SVE, and Advanced SIMD and floating-point functionality.

Configuration

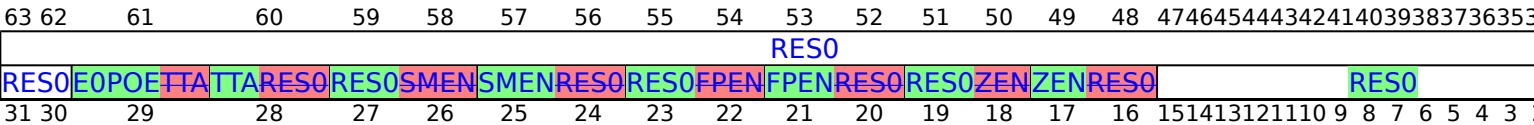
AArch64 System register CPACR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CPACR\[31:0\]](#).

When EL2 is implemented and enabled in the current Security state and [HCR_EL2](#).{E2H, TGE} == {1, 1}, the fields in this register have no effect on execution at EL0 and EL1. In this case, the controls provided by [CPTR_EL2](#) are used.

Attributes

CPACR_EL1 is a 64-bit register.

Field descriptions



Bits [63:3029]

Reserved, RES0.

EOPOE, bit [29]

When FEAT_S1POE is implemented:

Enable access to [POR_EL0](#).

Traps EL0 accesses to [POR_EL0](#), from AArch64 state only to EL1, or to EL2 when it is implemented and enabled in the current Security state and [HCR_EL2](#).TGE is 1. The exception is reported using ESR_ELx.EC value 0x18.

EOPOE	Meaning
0b0	This control causes EL0 access to POR_EL0 to be trapped.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TTA, bit [28]

Traps EL0 and EL1 System register accesses to all implemented trace registers from both Execution states to EL1, or to EL2 when it is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) is 1, as follows:

- In AArch64 state, accesses to trace registers are trapped, reported using [ESR_ELx.EC](#) value 0x18.
- In AArch32 state, MRC and MCR accesses to trace registers are trapped, reported using [ESR_ELx.EC](#) value 0x05.
- In AArch32 state, MRRC and MCRR accesses to trace registers are trapped, reported using [ESR_ELx.EC](#) value 0x0C.

TTA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	This control causes EL0 and EL1 System register accesses to all implemented trace registers to be trapped.

Note

- The ETMv4 architecture and ETE do not permit EL0 to access the trace registers. If the trace unit implements FEAT_ETMv4 or FEAT_ETE, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPACR_EL1.TTA](#) is 1.
- The Arm architecture does not provide traps on trace register accesses through the optional memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not implemented, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

SMEN, bits [25:24]**When FEAT_SME is implemented:**

Traps execution at EL1 and EL0 of SME instructions, SVE instructions when FEAT_SVE is not implemented or the PE is in Streaming SVE mode, and instructions that directly access the [SVCR](#) or [SMCR_EL1](#) System registers to EL1, or to EL2 when EL2 is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) is 1.

When instructions that directly access the [SVCR](#) System register are trapped with reference to this control, the MSR_SVCRSM, MSR_SVCRZA, and MSR_SVCRSMZA instructions are also trapped.

The exception is reported using [ESR_ELx.EC](#) value of 0x1D, with an ISS code of 0x00000000.

This field does not affect whether Streaming SVE or SME register values are valid.

A trap taken as a result of CPACR_EL1.SMEN has precedence over a trap taken as a result of CPACR_EL1.FPEN.

SMEN	Meaning
0b00	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b01	This control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL1 to be trapped.
0b10	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [23:22]

Reserved, RES0.

FPEN, bits [21:20]

Traps execution at EL1 and EL0 of instructions that access the Advanced SIMD and floating-point registers from both Execution states to EL1, reported using ESR_ELx.EC value 0x07, or to EL2 reported using ESR_ELx.EC value 0x00 when EL2 is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) is 1, as follows:

- In AArch64 state, accesses to [FPCR](#), [FPSR](#), any of the SIMD and floating-point registers V0-V31, including their views as D0-D31 registers or S0-31 registers.
- In AArch32 state, [FPSCR](#), and any of the SIMD and floating-point registers Q0-15, including their views as D0-D31 registers or S0-31 registers.

Traps execution at EL1 and EL0 of SME and SVE instructions to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1. The exception is reported using ESR_ELx.EC value 0x07.

A trap taken as a result of CPACR_EL1.SMEN has precedence over a trap taken as a result of CPACR_EL1.FPEN.

A trap taken as a result of CPACR_EL1.ZEN has precedence over a trap taken as a result of CPACR_EL1.FPEN.

FPEN	Meaning
0b00	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b01	This control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL1 to be trapped.
0b10	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

Writes to [MVFR0](#), [MVFR1](#), and [MVFR2](#) from EL1 or higher are CONSTRAINED UNPREDICTABLE and whether these accesses can be trapped by this control depends on implemented CONSTRAINED UNPREDICTABLE behavior.

Note

- Attempts to write to the FPSID count as use of the registers for accesses from EL1 or higher.
- Accesses from EL0 to [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), and [FPEXC](#) are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of CPACR_EL1.FPEN is not 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:18]

Reserved, RES0.

ZEN, bits [17:16]**When FEAT_SVE is implemented:**

Traps execution at EL1 and EL0 of SVE instructions when the PE is not in Streaming SVE mode, and instructions that directly access the [ZCR_EL1](#) System register to EL1, or to EL2 when EL2 is implemented and enabled in the current Security state and [HCR_EL2](#).TGE is 1.

The exception is reported using ESR_ELx.EC value 0x19.

A trap taken as a result of CPACR_EL1.ZEN has precedence over a trap taken as a result of CPACR_EL1.FPEN.

ZEN	Meaning
0b00	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b01	This control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL1 to be trapped.
0b10	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [15:0]

Reserved, RES0.

Accessing CPACR_EL1

When [HCR_EL2](#).E2H is 1, without explicit synchronization, access from EL3 using the mnemonic CPACR_EL1 or CPACR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CPACR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.CPACR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x100];
    else
        X[t, 64] = CPACR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = CPTR_EL2;
    else
        X[t, 64] = CPACR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CPACR_EL1;

```

MSR CPACR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.CPACR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x100] = X[t, 64];
    else
        CPACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        CPTR_EL2 = X[t, 64];
    else
        CPACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CPACR_EL1 = X[t, 64];

```

MRS <Xt>, CPACR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x100];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = CPACR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = CPACR_EL1;
    else
        UNDEFINED;

```

MSR CPACR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x100] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            CPACR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        CPACR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPP RCTX, Cache Prefetch Prediction Restriction by Context

The CPP RCTX characteristics are:

Purpose

Cache Prefetch Prediction Restriction by Context applies to all Cache Allocation Resources that predict cache allocations based on information gathered within the target execution context or contexts.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control cache prefetch predictions occurring after the instruction is complete and synchronized.

This instruction applies to all:

- Instruction caches.
- Data caches.
- TLB prefetching hardware used by the executing PE that applies to the supplied context or contexts.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to CPP RCTX are UNDEFINED.

Attributes

CPP RCTX is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0															GVMID	VMID															
RES0					NSE	NS	EL	RES0							GASID	ASID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 and EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]**When FEAT_RME is implemented:**

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see [CPP_RCTX.NS](#).

Otherwise:

Reserved, RES0.

NS, bit [26]**When FEAT_RME is implemented:**

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.
- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.
- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

This instruction is treated as a NOP when executed at EL3 and either:

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with CPP_RCTX.{NSE, NS} == {1, 0}.

- CPP_RCTX.{NSE, NS} selects a reserved value.
- CPP_RCTX.{NSE, NS} == {1, 0} and CPP_RCTX.EL has a value other than 0b11.

Otherwise:

Security State. Defined values are:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Executing the CPP RCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

CPP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b111

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.EnRCTX == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.CPPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPTR_EL2, Architectural Feature Trap Register (EL2)

The CPT_R_EL2 characteristics are:

Purpose

Controls trapping to EL2 of accesses to [CPACR](#), [CPACR_EL1](#), trace, Activity Monitor, SME, Streaming SVE, SVE, and Advanced SIMD and floating-point functionality.

Configuration

AArch64 System register CPT_R_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HCPTR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

CPTR_EL2 is a 64-bit register.

Field descriptions

When FEAT_VHE is implemented and HCR_EL2.E2H == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
TCPAC	TAM	EOP	RES0	TTA	RES0	SMEN	RES0	FPEN	RES0	ZEN	RES0																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

TCPAC, bit [31]

In AArch64 state, traps accesses to [CPACR_EL1](#) from EL1 to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x18.

In AArch32 state, traps accesses to [CPACR](#) from EL1 to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x03.

TCPAC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to CPACR_EL1 and CPACR are trapped to EL2, when EL2 is enabled in the current Security state.

When [HCR_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

Note

[CPACR_EL1](#) and [CPACR](#) are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TAM, bit [30]**When FEAT_AMUv1 is implemented:**

Trap Activity Monitor access. Traps EL1 and EL0 accesses to all Activity Monitor registers to EL2, as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using ESR_ELx.EC value 0x18:
 - [AMUSERENR_EL0](#), [AMCFGR_EL0](#), [AMCGCR_EL0](#), [AMCNTENCLR0_EL0](#), [AMCNTENCLR1_EL0](#), [AMCNTENSET0_EL0](#), [AMCNTENSET1_EL0](#), [AMCR_EL0](#), [AMEVCNTR0<n>_EL0](#), [AMEVCNTR1<n>_EL0](#), [AMEVTYPER0<n>_EL0](#), and [AMEVTYPER1<n>_EL0](#).
- In AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2 and reported using ESR_ELx.EC value 0x03:
 - [AMUSERENR](#), [AMCFGR](#), [AMCGCR](#), [AMCNTENCLR0](#), [AMCNTENCLR1](#), [AMCNTENSET0](#), [AMCNTENSET1](#), [AMCR](#), [AMEVTYPER0<n>](#), and [AMEVTYPER1<n>](#).
- In AArch32 state, MRRC or MCRR accesses to [AMEVCNTR0<n>](#) and [AMEVCNTR1<n>](#), are trapped to EL2, reported using ESR_ELx.EC value 0x04.

TAM	Meaning
0b0	Accesses from EL1 and EL0 to Activity Monitor registers are not trapped.
0b1	Accesses from EL1 and EL0 to Activity Monitor registers are trapped to EL2, when EL2 is enabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E0POE, bit [29]**When FEAT_S1POE is implemented:**

Enable access to [POR_EL0](#).

Traps EL0 accesses to [POR_EL0](#) to EL2, from AArch64 state only. The exception is reported using ESR_ELx.EC value 0x18.

E0POE	Meaning
0b0	This control causes EL0 access to POR_EL0 to be trapped.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TTA, bit [28]

Traps System register accesses to all implemented trace registers from both Execution states to EL2, when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to trace registers with op0=2, op1=1, and CRn<0b1000 are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, MRC or MCR accesses to trace registers with cpnum=14, opc1=1, and CRn<0b1000 are trapped to EL2, reported using EC syndrome value 0x05.

TTA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt at EL0, EL1 or EL2, to execute a System register access to an implemented trace register is trapped to EL2, when EL2 is enabled in the current Security state, unless HCR_EL2.TGE is 0 and it is trapped by CPACR.NSTRCDIS or CPACR_EL1.TTA . When HCR_EL2.TGE is 1, any attempt at EL0 or EL2 to execute a System register access to an implemented trace register is trapped to EL2, when EL2 is enabled in the current Security state.

Note

The ETMv4 architecture and ETE do not permit EL0 to access the trace registers. If the trace unit implements FEAT_ETMv4 or ETE, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of CPTR_EL2.TTA is 1.

EL2 does not provide traps on trace register accesses through the optional Memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

SMEN, bits [25:24]

When FEAT_SME is implemented:

Traps execution at EL2, EL1, and EL0 of SME instructions, SVE instructions when FEAT_SVE is not implemented or the PE is in Streaming SVE mode, and instructions that directly access the [SVCR](#), [SMCR_EL1](#), or [SMCR_EL2](#) System registers to EL2, when EL2 is enabled in the current Security state.

When instructions that directly access the [SVCR](#) System register are trapped with reference to this control, the MSR_SVCRSM, MSR_SVCRZA, and MSR_SVCRSMZA instructions are also trapped.

The exception is reported using ESR_EL2.EC value of 0x1D, with an ISS code of 0x0000000.

This field does not affect whether Streaming SVE or SME register values are valid.

A trap taken as a result of CPTR_EL2.SMEN has precedence over a trap taken as a result of CPTR_EL2.FPEN.

SMEN	Meaning
0b00	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b01	When HCR_EL2.TGE is 0, this control does not cause execution of any instructions to be trapped. When HCR_EL2.TGE is 1, this control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL2 to be trapped.
0b10	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [23:22]

Reserved, RES0.

FPEN, bits [21:20]

Traps execution at EL2, EL1, and EL0 of instructions that access the Advanced SIMD and floating-point registers from both Execution states to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x07.

Traps execution at EL2, EL1, and EL0 of SME and SVE instructions to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x07.

A trap taken as a result of CPTR_EL2.SMEN has precedence over a trap taken as a result of CPTR_EL2.FPEN.

A trap taken as a result of CPTR_EL2.ZEN has precedence over a trap taken as a result of CPTR_EL2.FPEN.

FPEN	Meaning
0b00	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b01	When HCR_EL2.TGE is 0, this control does not cause execution of any instructions to be trapped. When HCR_EL2.TGE is 1, this control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL2 to be trapped.
0b10	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

Writes to [MVFR0](#), [MVFR1](#), and [MVFR2](#) from EL1 or higher are CONSTRAINED UNPREDICTABLE and whether these accesses can be trapped by this control depends on implemented CONSTRAINED UNPREDICTABLE behavior.

Note

- Attempts to write to the FPSID count as use of the registers for accesses from EL1 or higher.
- Accesses from EL0 to [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), and [FPEXC](#) are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of CPTR_EL2.FPEN is not 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:18]

Reserved, RES0.

ZEN, bits [17:16]**When FEAT_SVE is implemented:**

Traps execution at EL2, EL1, and EL0 of SVE instructions when the PE is not in Streaming SVE mode, and instructions that directly access the [ZCR_EL1](#) or [ZCR_EL2](#) System registers to EL2, when EL2 is enabled in the current Security state.

The exception is reported using ESR_ELx.EC value 0x19.

A trap taken as a result of CPTR_EL2.ZEN has precedence over a trap taken as a result of CPTR_EL2.FPEN.

ZEN	Meaning
0b00	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b01	When HCR_EL2.TGE is 0, this control does not cause execution of any instructions to be trapped. When HCR_EL2.TGE is 1, this control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL2 to be trapped.
0b10	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b11	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [15:0]

Reserved, RES0.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32														
RES0																																													
TCPAC		TAM		RES0																	TTA		RES0					RES1		TSM	RES0	TFP	RES1	TZ		RES1									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														

This format applies in all Armv8.0 implementations.

Bits [63:32]

Reserved, RES0.

TCPAC, bit [31]

In AArch64 state, traps accesses to [CPACR_EL1](#) from EL1 to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x18.

In AArch32 state, traps accesses to [CPACR](#) from EL1 to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x03.

TCPAC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to the following registers are trapped to EL2, when EL2 is enabled in the current Security state: <ul style="list-style-type: none"> • CPACR_EL1. • CPACR.

When [HCR_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

Note

[CPACR_EL1](#) and [CPACR](#) are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TAM, bit [30]

When FEAT_AMUv1 is implemented:

Trap Activity Monitor access. Traps EL1 and EL0 accesses to all Activity Monitor registers to EL2, as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using ESR_ELx.EC value 0x18:
 - [AMUSERENR_EL0](#), [AMCFGR_EL0](#), [AMCGCR_EL0](#), [AMCNTENCLR0_EL0](#), [AMCNTENCLR1_EL0](#), [AMCNTENSET0_EL0](#), [AMCNTENSET1_EL0](#), [AMCR_EL0](#), [AMEVCNTR0<n>_EL0](#), [AMEVCNTR1<n>_EL0](#), [AMEVTYPER0<n>_EL0](#), and [AMEVTYPER1<n>_EL0](#).
- In AArch32 state, MCR or MRC accesses to the following registers are trapped to EL2 and reported using ESR_ELx.EC value 0x03:
 - [AMUSERENR](#), [AMCFGR](#), [AMCGCR](#), [AMCNTENCLR0](#), [AMCNTENCLR1](#), [AMCNTENSET0](#), [AMCNTENSET1](#), [AMCR](#), [AMEVTYPER0<n>](#), and [AMEVTYPER1<n>](#).
- In AArch32 state, MCRR or MRRC accesses to [AMEVCNTR0<n>](#) and [AMEVCNTR1<n>](#), are trapped to EL2, reported using ESR_ELx.EC value 0x04.

TAM	Meaning
0b0	Accesses from EL1 and EL0 to Activity Monitor registers are not trapped.
0b1	Accesses from EL1 and EL0 to Activity Monitor registers are trapped to EL2, when EL2 is enabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:21]

Reserved, RES0.

TTA, bit [20]

Traps System register accesses to all implemented trace registers from both Execution states to EL2, when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to trace registers with op0=2, op1=1, and CRn<0b1000 are trapped to EL2, reported using EC syndrome value 0x18.

- In AArch32 state, MRC or MCR accesses to trace registers with cpnum=14, opc1=1, and CRn<0b1000 are trapped to EL2, reported using EC syndrome value 0x05.

TTA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt at EL0, EL1, or EL2, to execute a System register access to an implemented trace register is trapped to EL2, when EL2 is enabled in the current Security state, unless it is trapped by one of the following controls: <ul style="list-style-type: none"> • CPACR_EL1.TTA • CPACR.TRCDIS

Note

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the trace unit implements FEAT_ETMv4, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPTR_EL2.TTA](#) is 1.
- EL2 does not provide traps on trace register accesses through the optional memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:14]

Reserved, RES0.

Bit [13]

Reserved, RES1.

TSM, bit [12]**When FEAT_SME is implemented:**

Traps execution at EL2, EL1, and EL0 of SME instructions, SVE instructions when FEAT_SVE is not implemented or the PE is in Streaming SVE mode, and instructions that directly access the [SVCRR](#), [SMCR_EL1](#), or [SMCR_EL2](#) System registers to EL2, when EL2 is enabled in the current Security state.

When instructions that directly access the [SVCRR](#) System register are trapped with reference to this control, the MSR_SVCRSM, MSR_SVCRZA, and MSR_SVCRSMZA instructions are also trapped.

The exception is reported using ESR_EL2.EC value of 0x1D, with an ISS code of 0x0000000.

This field does not affect whether Streaming SVE or SME register values are valid.

A trap taken as a result of CPTR_EL2.TSM has precedence over a trap taken as a result of CPTR_EL2.TFP.

TSM	Meaning
0b0	This control does not cause execution of any instructions to be trapped.
0b1	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

Bit [11]

Reserved, RES0.

TFP, bit [10]

Traps execution of instructions which access the Advanced SIMD and floating-point functionality, from both Execution states to EL2, when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using ESR_ELx.EC value 0x07:
 - [FPCR](#), [FPSR](#), [FPEXC32_EL2](#), any of the SIMD and floating-point registers V0-V31, including their views as D0-D31 registers or S0-31 registers.
- In AArch32 state, accesses to the following registers are trapped to EL2, reported using ESR_ELx.EC value 0x07:
 - [MVFR0](#), [MVFR1](#), [MVFR2](#), [FPSCR](#), [FPEXC](#), and any of the SIMD and floating-point registers Q0-15, including their views as D0-D31 registers or S0-31 registers. For the purposes of this trap, the architecture defines a VMSR access to [FPSID](#) from EL1 or higher as an access to a SIMD and floating-point register. Otherwise, permitted VMSR accesses to [FPSID](#) are ignored.

Traps execution at the same Exception levels of SME and SVE instructions to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x07.

A trap taken as a result of CPTR_EL2.TSM has precedence over a trap taken as a result of CPTR_EL2.TFP.

A trap taken as a result of CPTR_EL2.TZ has precedence over a trap taken as a result of CPTR_EL2.TFP.

TFP	Meaning
0b0	This control does not cause execution of any instructions to be trapped.
0b1	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.

Note

[FPEXC32_EL2](#) is not accessible from EL0 using AArch64.

[FPSID](#), [MVFR0](#), [MVFR1](#), and [FPEXC](#) are not accessible from EL0 using AArch32.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [9]

Reserved, RES1.

TZ, bit [8]**When FEAT_SVE is implemented:**

Traps execution at EL2, EL1, and EL0 of SVE instructions when the PE is not in Streaming SVE mode, and instructions that directly access the [ZCR_EL2](#) or [ZCR_EL1](#) System registers to EL2, when EL2 is enabled in the current Security state.

The exception is reported using ESR_ELx.EC value 0x19.

A trap taken as a result of CPTR_EL2.TZ has precedence over a trap taken as a result of CPTR_EL2.TFP.

TZ	Meaning
0b0	This control does not cause execution of any instructions to be trapped.
0b1	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

Bits [7:0]

Reserved, RES1.

Accessing CPTR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CPTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = CPTR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CPTR_EL2;

```

MSR CPTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        CPTR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CPTR_EL2 = X[t, 64];

```

MRS <Xt>, CPACR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.CPACR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x100];
    else
        X[t, 64] = CPACR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = CPTR_EL2;
    else
        X[t, 64] = CPACR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CPACR_EL1;

```

MSR CPACR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif EL2Enabled() && CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.CPACR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x100] = X[t, 64];
    else
        CPACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TCPAC == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.TCPAC == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        CPTR_EL2 = X[t, 64];
    else
        CPACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CPACR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPTR_EL3, Architectural Feature Trap Register (EL3)

The CPT_R_EL3 characteristics are:

Purpose

Controls trapping to EL3 of accesses to [CPACR](#), [CPACR_EL1](#), [HCPTR](#), [CPTR_EL2](#), trace, Activity Monitor, SME, Streaming SVE, SVE, and Advanced SIMD and floating-point functionality.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to CPT_R_EL3 are UNDEFINED.

Attributes

CPT_R_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
RES0																																			
TCPAC	TAM	RES0										TTA	RES0								ESM	RES0	TFP	RES0	EZ	RES0									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:32]

Reserved, RES0.

TCPAC, bit [31]

Traps all of the following to EL3, from both Execution states and any Security state.

- EL2 accesses to [CPTR_EL2](#), reported using ESR_ELx.EC value 0x18, or [HCPTR](#), reported using ESR_ELx.EC value 0x03.
- EL2 and EL1 accesses to [CPACR_EL1](#) reported using ESR_ELx.EC value 0x18, or [CPACR](#) reported using ESR_ELx.EC value 0x03.

When CPT_R_EL3.TCPAC is:

TCPAC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL2 accesses to the CPTR_EL2 or HCPTR , and EL2 and EL1 accesses to the CPACR_EL1 or CPACR , are trapped to EL3, unless they are trapped by CPTR_EL2 .TCPAC.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TAM, bit [30]

When FEAT_AMUv1 is implemented:

Trap Activity Monitor access. Traps EL2, EL1, and EL0 accesses to all Activity Monitor registers to EL3.

Accesses to the Activity Monitors registers are trapped as follows:

- In AArch64 state, the following registers are trapped to EL3 and reported with ESR_ELx.EC value 0x18:
 - [AMUSERENR_EL0](#), [AMCFGR_EL0](#), [AMCGCR_EL0](#), [AMCNTENCLR0_EL0](#), [AMCNTENCLR1_EL0](#), [AMCNTENSET0_EL0](#), [AMCNTENSET1_EL0](#), [AMCR_EL0](#), [AMEVCNTR0<n>_EL0](#), [AMEVCNTR1<n>_EL0](#), [AMEVTYPER0<n>_EL0](#), and [AMEVTYPER1<n>_EL0](#).
- In AArch32 state, accesses with MRC or MCR to the following registers reported with ESR_ELx.EC value 0x03:
 - [AMUSERENR](#), [AMCFGR](#), [AMCGCR](#), [AMCNTENCLR0](#), [AMCNTENCLR1](#), [AMCNTENSET0](#), [AMCNTENSET1](#), [AMCR](#), [AMEVTYPER0<n>](#), and [AMEVTYPER1<n>](#).
- In AArch32 state, accesses with MRRC or MCRR to the following registers, reported with ESR_ELx.EC value 0x04:
 - [AMEVCNTR0<n>](#), [AMEVCNTR1<n>](#).

TAM	Meaning
0b0	Accesses from EL2, EL1, and EL0 to Activity Monitor registers are not trapped.
0b1	Accesses from EL2, EL1, and EL0 to Activity Monitor registers are trapped to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:21]

Reserved, RES0.

TTA, bit [20]

Traps System register accesses. Accesses to the trace registers, from all Exception levels, any Security state, and both Execution states are trapped to EL3 as follows:

- In AArch64 state, Trace registers with op0=2, op1=1, and CRn<0b1000 are trapped to EL3 and reported using EC syndrome value 0x18.
- In AArch32 state, accesses using MCR or MRC to the Trace registers with cpnum=14, opc1=1, and CRn<0b1000 are reported using EC syndrome value 0x05.

TTA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any System register access to the trace registers is trapped to EL3, unless it is trapped by CPACR.TRCDIS , CPACR_EL1.TTA , or CPTR_EL2.TTA .

If System register access to trace functionality is not supported, this bit is RES0.

Note

The ETMv4 architecture and ETE do not permit EL0 to access the trace registers. If the trace unit implements FEAT_ETMv4 or FEAT_ETE, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than this trap exception.

EL3 does not provide traps on trace register accesses through the Memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, no side-effects occur before the exception is taken, see '[Configurable Traps instructions controls instructions](#)'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:13]

Reserved, RES0.

ESM, bit [12]

When FEAT_SME is implemented:

Traps execution of SME instructions, SVE instructions when FEAT_SVE is not implemented or the PE is in Streaming SVE mode, and instructions that directly access the [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#), [SMPRI_EL1](#), [SMPRIMAP_EL2](#), or [SVCR](#) System registers, from all Exception levels and any Security state, to EL3.

When instructions that directly access the [SVCR](#) System register are trapped with reference to this control, the MSR SVCRSM, MSR SVCRZA, and MSR SVCRSMZA instructions are also trapped.

When direct accesses to [SMPRI_EL1](#) and [SMPRIMAP_EL2](#) are trapped, the exception is reported using an [ESR_EL3](#).EC value of 0x18. Otherwise, the exception is reported using an [ESR_EL3](#).EC value of 0x1D, with an ISS code of 0x0000000.

This field does not affect whether Streaming SVE or SME register values are valid.

A trap taken as a result of CPTR_EL3.ESM has precedence over a trap taken as a result of CPTR_EL3.TFP.

ESM	Meaning
0b0	This control causes execution of these instructions at all Exception levels to be trapped.
0b1	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [11]

Reserved, RES0.

TFP, bit [10]

Traps execution of instructions which access the Advanced SIMD and floating-point functionality, from all Exception levels, any Security state, and both Execution states, to EL3.

This includes the following registers, all reported using ESR_ELx.EC value 0x07:

- [FPCR](#), [FPSR](#), [FPEXC32_EL2](#), and any of the SIMD and floating-point registers V0-V31, including their views as D0-D31 registers or S0-S31 registers.
- [MVFR0](#), [MVFR1](#), [MVFR2](#), [FPSCR](#), [FPEXC](#), and any of the SIMD and floating-point registers Q0-Q15, including their views as D0-D31 registers or S0-S31 registers.
- VMSR accesses to [FPSID](#).

Permitted VMSR accesses to [FPSID](#) are ignored, but for the purposes of this trap the architecture defines a VMSR access to the [FPSID](#) from EL1 or higher as an access to a SIMD and floating-point register.

Traps execution at all Exception levels of SME and SVE instructions to EL3 from any Security state. The exception is reported using ESR_ELx.EC value 0x07.

A trap taken as a result of CPTR_EL3.ESM has precedence over a trap taken as a result of CPTR_EL3.TFP.

A trap taken as a result of CPTR_EL3.EZ has precedence over a trap taken as a result of CPTR_EL3.TFP.

Defined values are:

TFP	Meaning
0b0	This control does not cause execution of any instructions to be trapped.
0b1	This control causes execution of these instructions at all Exception levels to be trapped.

Note

[FPEXC32_EL2](#) is not accessible from EL0 using AArch64.

[FPSID](#), [MVFR0](#), [MVFR1](#), and [FPEXC](#) are not accessible from EL0 using AArch32.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [9]

Reserved, RES0.

EZ, bit [8]

When FEAT_SVE is implemented:

Traps execution of SVE instructions when the PE is not in Streaming SVE mode, and instructions that directly access the [ZCR_EL3](#), [ZCR_EL2](#), or [ZCR_EL1](#) System registers, from all Exception levels and any Security state, to EL3.

The exception is reported using ESR_ELx.EC value 0x19.

A trap taken as a result of CPTR_EL3.EZ has precedence over a trap taken as a result of CPTR_EL3.TFP.

EZ	Meaning
0b0	This control causes execution of these instructions at all Exception levels to be trapped.
0b1	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [7:0]

Reserved, RES0.

Accessing CPTR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CPTR_EL3

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b110	0b0001	0b0001	0b010
------	-------	--------	--------	-------

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CPTR_EL3;
```

MSR CPTR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0001	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    CPTR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

CSSELR_EL1, Cache Size Selection Register

The CSSELR_EL1 characteristics are:

Purpose

Selects the current Cache Size ID Register, [CCSIDR_EL1](#), by specifying the required cache level and the cache type (either instruction or data cache).

Configuration

AArch64 System register CSSELR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CSSELR\[31:0\]](#).

Attributes

CSSELR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:5]

Reserved, RES0.

TnD, bit [4]

When FEAT_MTE2 is implemented:

Allocation Tag not Data bit.

TnD	Meaning
0b0	Data, Instruction or Unified cache.
0b1	Separate Allocation Tag cache.

When CSSELR_EL1.InD == 1, this bit is RES0.

If CSSELR_EL1.{TnD, Level, InD} is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR_EL1 is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Level, bits [3:1]

Cache level of required cache.

Level	Meaning
0b000	Level 1 cache.
0b001	Level 2 cache.
0b010	Level 3 cache.
0b011	Level 4 cache.
0b100	Level 5 cache.
0b101	Level 6 cache.
0b110	Level 7 cache.

All other values are reserved.

If CSSELR_EL1.{TnD, Level, InD} is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR_EL1 is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [0]

Instruction not Data bit.

InD	Meaning
0b0	Data or unified cache.
0b1	Instruction cache.

If CSSELR_EL1.{TnD, Level, InD} is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 is CONSTRAINED UNPREDICTABLE, and returns UNKNOWN values for CSSELR_EL1.{Level, InD}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing CSSELR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CSSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.CSSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CSSELR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CSSELR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CSSELR_EL1;

```

MSR CSSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.CSSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CSSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    CSSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CSSELR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CTR_EL0, Cache Type Register

The CTR_EL0 characteristics are:

Purpose

Provides information about the architecture of the caches.

Configuration

AArch64 System register CTR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [CTR\[31:0\]](#).

Attributes

CTR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																								TminLine							
RES1	RES0	DIC	IDC	CWG				ERG				DminLine				L1lp	RES0								lminLine						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:38]

Reserved, RES0.

TminLine, bits [37:32]

When FEAT_MTE2 is implemented:

Tag minimum Line. Log₂ of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE.

Note

- For an implementation with cache lines containing 64 bytes of data and 4 Allocation Tags, this will be $\log_2(64/4) = 4$.
- For an implementation with Allocations Tags in separate cache lines of 128 Allocation Tags per line, this will be $\log_2(128*16/4) = 9$.

Otherwise:

Reserved, RES0.

Bit [31]

Reserved, RES1.

Bit [30]

Reserved, RES0.

DIC, bit [29]

Instruction cache invalidation requirements for data to instruction coherence.

DIC	Meaning
0b0	Instruction cache invalidation to the Point of Unification is required for data to instruction coherence.
0b1	Instruction cache invalidation to the Point of Unification is not required for data to instruction coherence.

IDC, bit [28]

Data cache clean requirements for instruction to data coherence. The meaning of this bit is:

IDC	Meaning
0b0	Data cache clean to the Point of Unification is required for instruction to data coherence, unless CLIDR_EL1.LoC == 0b000 or (CLIDR_EL1.LoUIS == 0b000 && CLIDR_EL1.LoUU == 0b000).
0b1	Data cache clean to the Point of Unification is not required for instruction to data coherence.

CWG, bits [27:24]

Cache writeback granule. Log₂ of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified.

A value of 0b0000 indicates that this register does not provide Cache writeback granule information and either:

- The architectural maximum of 512 words (2KB) must be assumed.
- The Cache writeback granule can be determined from maximum cache line size encoded in the Cache Size ID Registers.

Values greater than 0b1001 are reserved.

Arm recommends that an implementation that does not support cache write-back implements this field as 0b0001. This applies, for example, to an implementation that supports only write-through caches.

ERG, bits [23:20]

Exclusives reservation granule, and, if FEAT_TME is implemented, transactional reservation granule. Log₂ of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions, and, if FEAT_TME is implemented, for detecting transactional conflicts.

A value of 0b0000 indicates that this register does not provide granule information and the architectural maximum of 512 words (2KB) must be assumed.

Value 0b0001 and values greater than 0b1001 are reserved.

DminLine, bits [19:16]

Log₂ of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE.

L1Ip, bits [15:14]

Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are:

L1Ip	Meaning	Applies when
0b00	VMID aware Physical Index, Physical tag (VPIPT).	When FEAT_VPIPT is implemented
0b01	ASID-tagged Virtual Index, Virtual Tag (AIVIVT).	
0b10	Virtual Index, Physical Tag (VIPT).	
0b11	Physical Index, Physical Tag (PIPT).	

From Armv8, the value 0b01 is reserved.

The value 0b00 is permitted only in an implementation that includes FEAT_VPIPT.

Bits [13:4]

Reserved, RES0.

IminLine, bits [3:0]

Log₂ of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE.

Accessing CTR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, CTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCT == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.CTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CTR_EL0;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.CTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CTR_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = CTR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = CTR_EL0;

```

(old)**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

DBGAUTHSTATUS_EL1, Debug Authentication Status register

The DBGAUTHSTATUS_EL1 characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

Configuration

AArch64 System register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGAUTHSTATUS\[31:0\]](#).

AArch64 System register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to External register [DBGAUTHSTATUS_EL1\[31:0\]](#).

Attributes

DBGAUTHSTATUS_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0				RTNID		RTID		RES0								RLNID		RLID		RES0				SNID		SID		NSNID		NSID	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTID.

RTID, bits [25:24]

Root invasive debug.

RTID	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRootInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRootInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is 00. 0b00.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLID.

RLID, bits [13:12]

Realm invasive debug.

RLID	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRealmInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRealmInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is ~~00~~, 0b00.

Bits [11:8]

Reserved, RES0.

SNID, bits [7:6]**When FEAT_Debugv8p4 is implemented:**

Secure non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.SID.

Otherwise:

Secure non-invasive debug.

SNID	Meaning
0b00	Not implemented. One EL3 is not implemented and the following Effective is value true: of SCR_EL3.NS is 1. <ul style="list-style-type: none"> EL3 is not implemented and the Effective value of SCR_EL3.NS is 1. FEAT_RME is implemented without Secure state.
0b10	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

SID, bits [5:4]

Secure invasive debug.

SID	Meaning
0b00	Not implemented. One EL3 of is not implemented and the following Effective is value true: of SCR_EL3.NS is 1. <ul style="list-style-type: none"> EL3 is not implemented and the Effective value of SCR_EL3.NS is 1. FEAT_RME is implemented without Secure state.
0b10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSNID, bits [3:2]

When FEAT_Debugv8p4 is implemented:

Non-secure non-invasive debug.

NSNID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b11	Implemented and enabled. EL3 is implemented or the Effective value of SCR_EL3.NS is 1.

All other values are reserved.

Otherwise:

Non-secure non-invasive debug.

NSNID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSID, bits [1:0]

Non-secure invasive debug.

NSID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Accessing DBGAUTHSTATUS_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGAUTHSTATUS_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1110	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGAUTHSTATUS_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = DBGAUTHSTATUS_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = DBGAUTHSTATUS_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = DBGAUTHSTATUS_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n>_EL1 characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register [DBGBVR<n>_EL1](#).

Configuration

AArch64 System register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGBCR<n>\[31:0\]](#).

AArch64 System register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to External register [DBGBCR<n>_EL1\[31:0\]](#).

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

DBGBCR<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
LBNX	RES0	SSCE	MASK	RES0	BT				LBN			SSC	HMC	RES0			BAS			RES0	BT2	PMC	PMCE	E							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32:30]

Reserved, RES0.

LBNX, bits [31:30]

When FEAT_Debugv8p9 is implemented:

Linked Breakpoint Number.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

For all other breakpoint types, this field is ignored and reads of the register return an UNKNOWN value.

This field extends DBGBCR<n>_EL1.LBN to support up to 64 implemented breakpoints.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSCE, bit [29]**When FEAT_RME is implemented:**

Security State Control Extended.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MASK, bits [28:24]**When FEAT_ABLE is implemented:**

Address Mask. Only address ranges up to 2GB can be watched using a single mask.

MASK	Meaning
0b00000	No mask.
0b00001	Reserved.
0b00010	Reserved.
0b00011..0b11111	Number of address bits masked.

Indicates the number of masked address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type.

BT	Meaning	Applies when
0b0000	Unlinked instruction address match. DBGBVR<n>_EL1 is the address of an instruction.	
0b0001	Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	
0b0010	Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of HCR_EL2.E2H is 1, and if either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL1 value.	When breakpoint n is context-aware
0b0011	As 0b0010, with linking enabled.	When breakpoint n is context-aware
0b0100	Unlinked instruction address mismatch. CONTEXTIDR_EL1 match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 is the address of an instruction.	When FEAT_ABLE is implemented
0b0101	Linked instruction address mismatch. As 0b0100, but linked to a breakpoint that has linking enabled.	When FEAT_ABLE is implemented
0b0110	Unlinked VMID match. CONTEXTIDR_EL1 match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1.VTBR_EL2.VMID .	When FEAT_VHE is implemented and breakpoint n is context-aware
0b0111	As 0b0110, with linking enabled.	When FEAT_VHE is implemented and breakpoint n is context-aware
0b1000	Unlinked VMID and Context ID match. DBGBVR<n>_EL1.VMID is a VMID compared against CONTEXTIDR_EL1 , and DBGBVR<n>_EL1.VMID is a VMID compared against VTBR_EL2.VMID .	When EL2 is implemented and breakpoint n is context-aware
0b1001	As 0b1000, with linking enabled.	When EL2 is implemented and breakpoint n is context-aware
0b1010	Unlinked VMID and Context ID match. DBGBVR<n>_EL1.CONTEXTIDR_EL2 is a Context ID compared against CONTEXTIDR_EL1 , and DBGBVR<n>_EL1.VMID is a VMID compared against VTBR_EL2.CONTEXTIDR_EL2.VMID .	When EL2 is implemented and breakpoint n is context-aware
0b1011	As 0b1010, with linking enabled.	When EL2 is implemented and breakpoint n is context-aware
0b1100	Unlinked Full Context ID match. ContextID is compared against CONTEXTIDR_EL1.CONTEXTIDR_EL2 .	When FEAT_VHE is implemented

	match., and DBGBCR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2.	implemented and breakpoint n is context-aware. When FEAT_VHE is implemented and breakpoint n is context-aware. When FEAT_VHE is implemented and breakpoint n is context-aware.
0b11010b1111	As 0b11000b1110, with linking enabled.	
0b1110	Unlinked Full Context ID match. DBGBCR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBCR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2.	
0b1111	As 0b1110, with linking enabled.	

All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information on the effect of programming the fields to a reserved value, see 'Reserved DBGBCR<n>_EL1.BT values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked Breakpoint Number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value.

For This all other breakpoint types, this field is ignored and when read the value of the DBGBCR<n>_EL1.E register is return an UNKNOWN value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information on the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

When AArch32 is supported:

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0b0011	DBGBVR<n>_EL1	Use for T32 instructions
0b1100	DBGBVR<n>_EL1 + 2	Use for T32 instructions
0b1111	DBGBVR<n>_EL1	Use for A64 and A32 instructions

All other values are reserved. For more information, see 'Reserved DBGBCR<n>_EL1.BAS values'.

For more information on using the BAS field in address match breakpoints, see 'Using the BAS field in Address Match breakpoints'.

For Context matching breakpoints, this field is RES1 and ignored.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

BitBits [4:3]

Reserved, RES0.

BT2, bit [3]

When FEAT_ABLE is implemented:

Breakpoint Type 2. With DBGBCR<n>_EL1.BT, specifies breakpoint type.

BT2	Meaning
0b0	As DBGBCR<n>_EL1.BT.
0b1	As DBGBCR<n>_EL1.BT, but with linking enabled. This value is only defined for the following DBGBCR<n>_EL1.BT values: 0b0000, 0b0001, 0b0100, and 0b0101. All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable breakpoint [n.DBGBVR<n>_EL1](#).

E	Meaning
0b0	Breakpoint n disabled.
0b1	Breakpoint n enabled.

This bit is ignored by the PE and treated as zero when all of the following are true:

- FEAT_Debugv8p9 is implemented.
- Any of the following are true:
 - HaltOnBreakpointOrWatchpoint() is FALSE and the Effective value of [MDSCR_EL1.EBWE](#) is 0.
 - HaltOnBreakpointOrWatchpoint() is TRUE and the Effective value of [EDSCR2.EBWE](#) is 0.
- [n](#) > 16.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing DBGBCR<n>_EL1

When FEAT_Debugv8p9 is implemented, a PE is permitted to support up to 64 implemented breakpoints.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGBCR<m>_EL1 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b101

```
integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[m];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[m];
elseif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[m];
```

MSR DBGBCR<m>_EL1, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b101

```

integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            DBGBCR_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
                Halt(DebugHalt_SoftwareAccess);
            else
                DBGBCR_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            DBGBCR_EL1[m] = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 15

The DBGBVR<n>_EL1 characteristics are:

Purpose

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register [DBGBCR<n>_EL1](#).

Configuration

AArch64 System register DBGBVR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGBVR<n>\[31:0\]](#).

AArch64 System register DBGBVR<n>_EL1 bits [63:32] are architecturally mapped to AArch32 System register [DBGXVR<n>\[31:0\]](#).

AArch64 System register DBGBVR<n>_EL1 bits [63:0] are architecturally mapped to External register [DBGBVR<n>_EL1\[63:0\]](#).

How this register is interpreted depends on the value of [DBGBCR<n>_EL1](#).BT.

- When [DBGBCR<n>_EL1](#).BT is 0b000x, this register holds a virtual address.
- When [DBGBCR<n>_EL1](#).BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When [DBGBCR<n>_EL1](#).BT is 0b100x, this register holds a VMID.
- When [DBGBCR<n>_EL1](#).BT is 0b101x, this register holds a VMID and a Context ID.
- When [DBGBCR<n>_EL1](#).BT is 0b111x, this register holds two Context ID values.

For other values of [DBGBCR<n>_EL1](#).BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

DBGBVR<n>_EL1 is a 64-bit register.

Field descriptions

When DBGBCR<n>_EL1.BT == 0b000x:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32								
RESS[14:8]								RESS[14:4]				Bits[56:53]				Bits[52:49]				VA[48:2]																			
																				VA[48:2]																RES0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

RESS[14:84], bits [63:5753]

Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply:

- It is CONSTRAINED UNPREDICTABLE whether the PE ignores this field when comparing an address.
- If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.

Bits[56:53]

When FEAT_LVA3 is implemented:

VA[56:53], bits [3:0] of bits [56:53]

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RESS[7:4], bits [3:0] of bits [56:53]

Extension to RESS[14:8]. For more information, see RESS[14:8].

Bits[52:49]

When FEAT_LVA is implemented:

VA[52:49], bits [3:0] of bits [52:49]

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

RESS[3:0], bits [3:0] of bits [52:49]

Extension to RESS[14:84]. For more information, see RESS[14:84].

VA[48:2], bits [48:2]

Bits[48:2] of the address value for comparison.

When FEAT_LVA3 is implemented, (VA[56:53]:VA[52:49]) forms the upper part of the address value. Otherwise, bits [56:53] are part of the RESS field.

When FEAT_LVA is implemented, VA[52:49] forms the upper part of the address value. Otherwise, bits [52:49] are part of the RESS field.

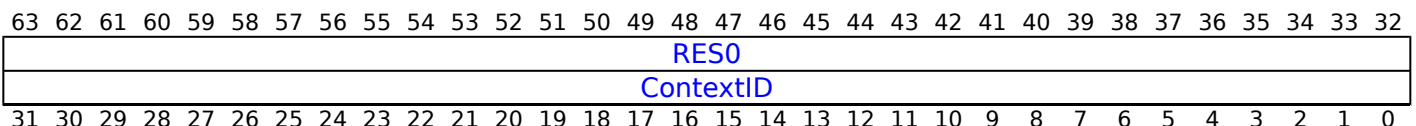
The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [1:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT == 0b001x:



Bits [63:32]

Reserved, RES0.

ContextID, bits [31:0]

Context ID value for comparison.

The value is compared against [CONTEXTIDR_EL2](#) when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), [HCR_EL2.E2H](#) is 1, and either:

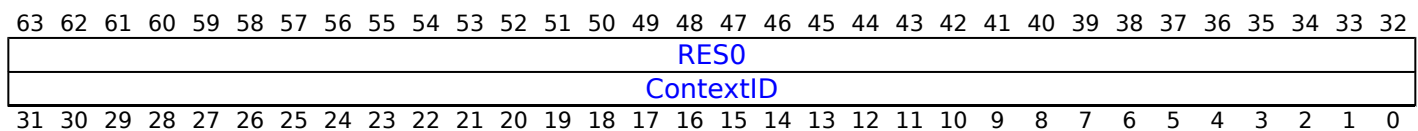
- The PE is executing at EL2.
- [HCR_EL2.TGE](#) is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state.

Otherwise, the value is compared against [CONTEXTIDR_EL1](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When DBGBCR<n>_EL1.BT == 0b011x:



Bits [63:32]

Reserved, RES0.

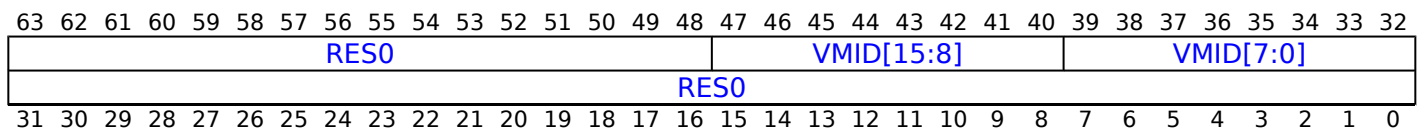
ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When DBGBCR<n>_EL1.BT == 0b100x and EL2 is implemented:



Bits [63:48]

Reserved, RES0.

VMID[15:8], bits [47:40]

When FEAT_VMID16 is implemented, VTCR_EL2.VS == 1 and EL2 is using AArch64:

Extension to VMID[7:0]. For more information, see [DBGBVR<n>_EL1.VMID\[7:0\]](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- [VTCR_EL2.VS](#) is 0.
- FEAT_VMID16 is not implemented.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [31:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT == 0b101x and EL2 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																VMID[15:8]								VMID[7:0]							
ContextID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

VMID[15:8], bits [47:40]

When FEAT_VMID16 is implemented, VTCR_EL2.VS == 1 and EL2 is using AArch64:

Extension to VMID[7:0]. For more information, see DBGBCR<n>_EL1.VMID[7:0].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- [VTCR_EL2.VS](#) is 0.
- FEAT_VMID16 is not implemented.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

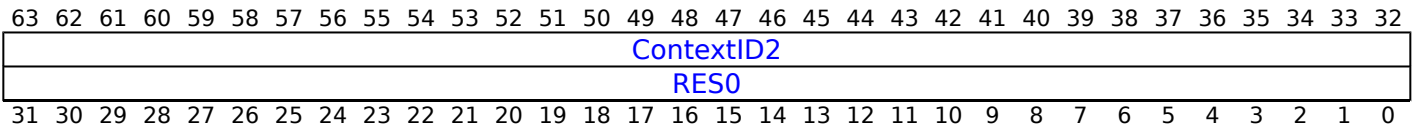
ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When DBGBCR<n>_EL1.BT == 0b110x, EL2 is implemented and (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented):



ContextID2, bits [63:32]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

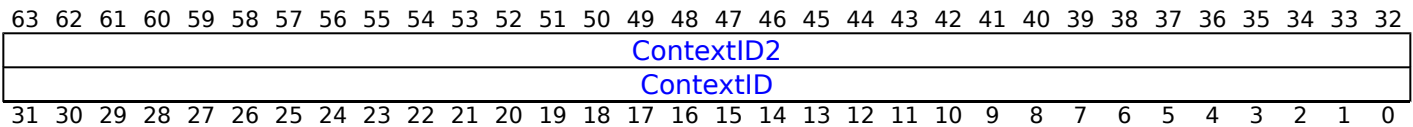
The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [31:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT == 0b111x, EL2 is implemented and (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented):



ContextID2, bits [63:32]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing DBGBVR<n>_EL1

When FEAT_Debugv8p9 is implemented, a PE is permitted to support up to 64 implemented breakpoints.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGBVR<m>_EL1 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b100

```
integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[m];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[m];
```

MSR DBGBVR<m>_EL1, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b100

```
integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            DBGBVR_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
                Halt(DebugHalt_SoftwareAccess);
            else
                DBGBVR_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            DBGBVR_EL1[m] = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear register

The DBGCLAIMCLR_EL1 characteristics are:

Purpose

Used by software to read the values of the CLAIM tag bits, and to clear CLAIM tag bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.

Note

CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the [DBGCLAIMSET_EL1](#) register.

Configuration

AArch64 System register DBGCLAIMCLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGCLAIMCLR\[31:0\]](#).

AArch64 System register DBGCLAIMCLR_EL1 bits [31:0] are architecturally mapped to External register [DBGCLAIMCLR_EL1\[31:0\]](#).

An implementation must include eight CLAIM tag bits.

Attributes

DBGCLAIMCLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32					
																RES0																				
																								CLAIM												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					

Bits [63:32]

Reserved, RES0.

Bits [31:8]

Reserved, RAZ/WI.

CLAIM, bits [7:0]

Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits.

Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0.

Writing 0 to one of these bits has no effect.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Accessing DBGCLAIMCLR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGCLAIMCLR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGCLAIMCLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGCLAIMCLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DBGCLAIMCLR_EL1;

```

MSR DBGCLAIMCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGCLAIMCLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGCLAIMCLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    DBGCLAIMCLR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGCLAIMSET_EL1, Debug CLAIM Tag Set register

The DBGCLAIMSET_EL1 characteristics are:

Purpose

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.

Note

CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the [DBGCLAIMCLR_EL1](#) register.

Configuration

AArch64 System register DBGCLAIMSET_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGCLAIMSET\[31:0\]](#).

AArch64 System register DBGCLAIMSET_EL1 bits [31:0] are architecturally mapped to External register [DBGCLAIMSET_EL1\[31:0\]](#).

An implementation must include eight CLAIM tag bits.

Attributes

DBGCLAIMSET_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
																RES0																			
RAZ/WI																								CLAIM											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:32]

Reserved, RES0.

Bits [31:8]

Reserved, RAZ/WI.

CLAIM, bits [7:0]

Set CLAIM tag bits.

This field is RAO.

Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1.

Writing 0 to one of these bits has no effect.

Accessing DBGCLAIMSET_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGCLAIMSET_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGCLAIMSET_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGCLAIMSET_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DBGCLAIMSET_EL1;

```

MSR DBGCLAIMSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGCLAIMSET_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGCLAIMSET_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    DBGCLAIMSET_EL1 = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.

On a Cold reset, if the powerup request is implemented and the powerup request has been asserted, this field is set to an IMPLEMENTATION DEFINED choice of 0 or 1. If the powerup request is not asserted, this field is set to 0.

Otherwise:

Core no powerdown request. Requests emulation of powerdown.

This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.

CORENPDRQ	Meaning
0b0	If the system responds to a powerdown request, it powers down Core power domain.
0b1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

In an implementation that includes the recommended external debug interface, this bit drives the DBGNOPWRDWN signal.

It is IMPLEMENTATION DEFINED whether this bit is reset to the value of [EDPRCR.COREPURQ](#) on exit from an IMPLEMENTATION DEFINED software-visible retention state. For more information about retention states see 'Core power domain power states'.

Note

Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.

The reset behavior of this field is:

- On a Cold reset, this field resets to the value in [EDPRCR.COREPURQ](#).

Accessing DBGPRCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGPRCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGPRCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGPRCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = DBGPRCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DBGPRCR_EL1;

```

MSR DBGPRCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGPRCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGPRCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        DBGPRCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    DBGPRCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n>_EL1 characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register [DBGWVR<n>_EL1](#).

Configuration

AArch64 System register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGWCR<n>\[31:0\]](#).

AArch64 System register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to External register [DBGWCR<n>_EL1\[31:0\]](#).

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

DBGWCR<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
LBNX	RES0	SSCE	MASK				RES0				WT	LBN				SSC	HMC	BAS								LSC	PAC	E			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32:30]

Reserved, RES0.

LBNX, bits [31:30]

When FEAT_Debugv8p9 is implemented:

Linked Breakpoint Number.

For Linked data address watchpoints, with DBGWCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.

This field extends DBGWCR<n>_EL1.LBN to support up to 64 implemented breakpoints.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSCE, bit [29]**When FEAT_RME is implemented:**

Security State Control Extended.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MASK, bits [28:24]

Address **Mask, mask**. Only objects up to 2GB can be watched using a single mask.

MASK	Meaning
0b00000	No mask.
0b00001	Reserved.
0b00010	Reserved.
0b00011..0b11111	Number of address bits masked.

Indicates the number of masked address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0b0	Unlinked data address match.
0b1	Linked data address match.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked ~~Breakpoint~~ ~~breakpoint~~ ~~Number~~ ~~number~~. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

For Linked data address watchpoints, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

For more information on the effect of programming the fields to a reserved value, see 'Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by [DBGWVR<n>_EL1](#) is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>_EL1
xxxxxx1x	Match byte at DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at DBGWVR<n>_EL1 + 2
xxx1xxx	Match byte at DBGWVR<n>_EL1 + 3

In cases where [DBGWVR<n>_EL1](#) addresses a double-word:

BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at DBGWVR<n>_EL1 + 7

If [DBGWVR<n>_EL1\[2\]](#) == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting [DBGWVR<n>_EL1\[2\]](#) == 1.

The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>_EL1.BAS values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
0b01	Match instructions that load from a watchpointed address.
0b10	Match instructions that store to a watchpointed address.
0b11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable watchpoint n. ~~Possible values are:~~

E	Meaning
0b0	Watchpoint n disabled.
0b1	Watchpoint n enabled.

This bit is ignored by the PE and treated as zero when all of the following are true:

- FEAT_Debugv8p9 is implemented.
- Any of the following are true:
 - HaltOnBreakpointOrWatchpoint() is FALSE and the Effective value of [MDSCR_EL1.EBWE](#) is 0.
 - HaltOnBreakpointOrWatchpoint() is TRUE and the Effective value of [EDSCR2.EBWE](#) is 0.
- n > 16.**

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing DBGWCR<n>_EL1

When FEAT_Debugv8p9 is implemented, a PE is permitted to support up to 64 implemented watchpoints.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGWCR<m>_EL1 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b111

```
integer m = UInt(CRm<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[m];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[m];
```

MSR DBGWCR<m>_EL1, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b111

```

integer m = UInt(CRm<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 15

The DBGWVR<n>_EL1 characteristics are:

Purpose

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register [DBGWCR<n>_EL1](#).

Configuration

AArch64 System register DBGWVR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGWVR<n>\[31:0\]](#).

AArch64 System register DBGWVR<n>_EL1 bits [63:0] are architecturally mapped to External register [DBGWVR<n>_EL1\[63:0\]](#).

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

DBGWVR<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
RESS[14:8]				RESS[14:4]				Bits[56:53]				Bits[52:49]				Bits[52:49]				VA[48:2]				VA[48:2]											
VA[48:2]																																RES0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

RESS[14:8], bits [63:57]

Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply:

- It is CONSTRAINED UNPREDICTABLE whether the PE ignores this field when comparing an address.
- It is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.

Bits[56:53]

When FEAT_LVA3 is implemented:

VA[56:53], bits [3:0] of bits [56:53]

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:**RESS[7:4], bits [3:0] of bits [56:53]**

Extension to RESS[14:8]. For more information, see RESS[14:8].

Bits[52:49]**When FEAT_LVA is implemented:****VA[52:49], bits [3:0] of bits [52:49]**

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:**RESS[3:0], bits [3:0] of bits [52:49]**

Extension to RESS[14:84]. For more information, see RESS[14:84].

VA[48:2], bits [48:2]

Bits[48:2] of the address value for comparison.

When FEAT_LVA3 is implemented, (VA[56:53]:VA[52:49]) forms the upper part of the address value. Otherwise, bits [56:53] are part of the RESS field.

When FEAT_LVA is implemented, VA[52:49] forms the upper part of the address value. Otherwise, bits [52:49] are part of the RESS field.

Arm deprecates setting `DBGWVR<n>_EL1[2] == 1`.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [1:0]

Reserved, RES0.

Accessing DBGWVR<n>_EL1

When FEAT_Debugv8p9 is implemented, a PE is permitted to support up to 64 implemented watchpoints.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DBGWVR<m>_EL1 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b110

```

integer m = UInt(CRm<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[m];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[m];

```

MSR DBGWVR<m>_EL1, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b110

```

integer m = UInt(CRM<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CGDSW, Clean of Data and Allocation Tags by Set/Way

The DC CGDSW characteristics are:

Purpose

Clean data and Allocation Tags in data cache by set/way.

Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CGDSW are UNDEFINED.

Attributes

DC CGDSW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																																
SetWay																															Level	RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

Executing the DC CGDSW instruction

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGDSW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.DCCSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_SetWay);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_SetWay);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_SetWay);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CGDVAC, Clean of Data and Allocation Tags by VA to PoC

The DC CGDVAC characteristics are:

Purpose

Clean data and Allocation Tags in data cache by address to Point of Coherency.

Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CGDVAC are UNDEFINED.

Attributes

DC CGDVAC is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
												VA Virtual address to use																				
												VA Virtual address to use																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

VA, bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGDVAC instruction

If EL0 access is enabled, when executed at EL0, this instruction may require read access permission to the VA, otherwise it generates a Permission fault, subject to the constraints described in 'MMU Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGDVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1010	0b101

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CGDVADP, Clean of Data and Allocation Tags by VA to PoDP

The DC CGDVADP characteristics are:

Purpose

Clean Allocation Tags and data in data cache by address to Point of Deep Persistence.

If the memory system does not identify a Point of Deep Persistence, then this instruction behaves as a [DC CGDVAP](#).

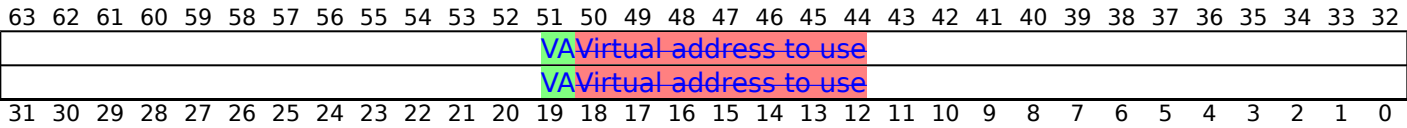
Configuration

This instruction is present only when FEAT_DPB2 is implemented and FEAT_MTE is implemented. Otherwise, direct accesses to DC CGDVADP are UNDEFINED.

Attributes

DC CGDVADP is a 64-bit System instruction.

Field descriptions



VA, bitsBits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGDVADP instruction

If ELO access is enabled, when executed at EL0, this instruction may requires generateread access Permissionpermission faultto the VA, subjectotherwise to it thegenerates constraintsa describedPermission infault, see 'MMUPermission faults generated by cache maintenance operationsfault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGDVADP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1101	0b101

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoDP);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

DC CGDVAP, Clean of Data and Allocation Tags by VA to PoP

The DC CGDVAP characteristics are:

Purpose

Clean data and Allocation Tags in data cache by address to Point of Persistence.
If the memory system does not identify a Point of Persistence, then this instruction behaves as a [DC CGDVAC](#).

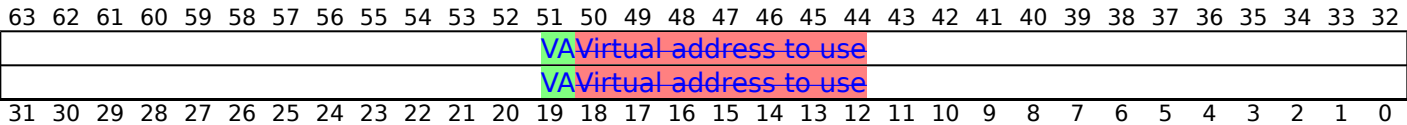
Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CGDVAP are UNDEFINED.

Attributes

DC CGDVAP is a 64-bit System instruction.

Field descriptions



VA, bitsBits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGDVAP instruction

If ELO access is enabled, when executed at EL0, this instruction may requires generateread access Permissionpermission faultto the VA, subjectotherwise to it thegenerates constraintsa describedPermission infault, see 'MMUPermission faults generated by cache maintenance operationsfault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGDVAP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1100	0b101

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Clean, CacheOpScope_PoP);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CGSW, Clean of Allocation Tags by Set/Way

The DC CGSW characteristics are:

Purpose

Clean Allocation Tags in data cache by set/way.

Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CGSW are UNDEFINED.

Attributes

DC CGSW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																SetWay												Level			RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1010	0b100

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

DC CGVAC, Clean of Allocation Tags by VA to PoC

The DC CGVAC characteristics are:

Purpose

Clean Allocation Tags in data cache by address to Point of Coherency.

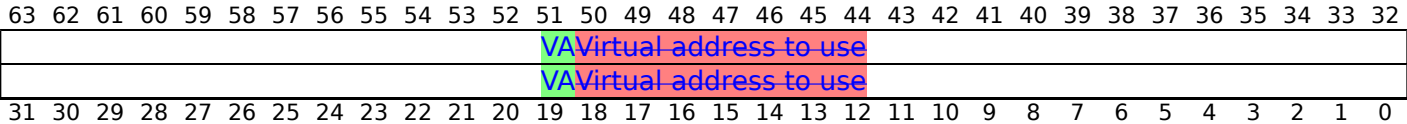
Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CGVAC are UNDEFINED.

Attributes

DC CGVAC is a 64-bit System instruction.

Field descriptions



VA, bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGVAC instruction

If EL0 access is enabled, when executed at EL0, this instruction requires read access permission to the VA, otherwise it generates a Permission fault, subject to the constraints described in 'MMU Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1010	0b011

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The DC CGVADP characteristics are:

Purpose

Clean Allocation tags by address to Point of Deep Persistence.

If the memory system does not identify a Point of Deep Persistence, then this instruction behaves as a [DC CGVAP](#).

Configuration

This instruction is present only when FEAT_DPB2 is implemented and FEAT_MTE is implemented. Otherwise, direct accesses to DC CGVADP are UNDEFINED.

Attributes

DC CGVADP is a 64-bit System instruction.

Field descriptions

Diagram illustrating the structure of a 64-bit virtual address, divided into three fields:

- Virtual Address Space Number (VSN):** 19 bits (bits 63 down to 45).
- Virtual Address to use:** 12 bits (bits 44 down to 33).
- Offset:** 33 bits (bits 32 down to 0).

VA, bits ~~Bits~~ [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGVADP instruction

If EL0 access is enabled, when executed at EL0, this instruction may require generate read access Permission permission fault to the VA, subject otherwise to the generates constraints a described Permission in fault, see 'MMU Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGVADP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1101	0b011

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoDP);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CGVAP, Clean of Allocation Tags by VA to PoP

The DC CGVAP characteristics are:

Purpose

Clean Allocation Tags in data cache by address to Point of Persistence.

If the memory system does not identify a Point of Persistence, then this instruction behaves as a [DC CGVAC](#).

Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CGVAP are UNDEFINED.

Attributes

DC CGVAP is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
												VA Virtual address to use																			
												VA Virtual address to use																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VA, bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CGVAP instruction

If EL0 access is enabled, when executed at EL0, this instruction may require a read access permission to the VA, subject to the constraints described in 'MMU Permission faults generated by cache maintenance operations fault'. Otherwise, it generates a fault, see 'Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CGVAP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1100	0b011

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Clean, CacheOpScope_PoP);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

DC CIGDPAE, Clean and invalidate of data and allocation tags by PA to PoE

The DC CIGDPAE characteristics are:

Purpose

Clean and invalidate of data and allocation tags by PA to PoE.

Configuration

This instruction is present only when FEAT_MEC is implemented. Otherwise, direct accesses to DC CIGDPAE are UNDEFINED.

Attributes

DC CIGDPAE is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Reserved, RES0.

Executing DC CIGDPAE

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIGDPAE, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoE);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoE);

```

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CIGDPAPA, Clean and Invalidate of Data and Allocation Tags by PA to PoPA

The DC CIGDPAPA characteristics are:

Purpose

Clean and Invalidate data and Allocation Tags in data cache by physical address to the Point of Physical Aliasing.

Note

This instruction cleans and invalidates all copies of the Location specified in the Xt argument, irrespective of any MECID associated with the Location. Memory accesses resulting from the Clean operation use the MECID associated with the cache entry.

Configuration

This instruction is present only when FEAT_RME is implemented and FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CIGDPAPA are UNDEFINED.

Attributes

DC CIGDPAPA is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
NS		NSE		RES0										Physical address																			
Physical address																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

NS, bit [63]

Together with the NSE field, this field specifies the target physical address space.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

If FEAT_SEL2 is not implemented, and {NSE, NS} == {0b0, 0b0}, then no cache entries are required to be cleaned or invalidated

NSE, bit [62]

Together with the NS field, this field specifies the target physical address space.

For a description of the values derived by evaluating NS and NSE together, see DC CIGDPAPA.NS.

(old)

htmldiff from-

(new)

DC CIGDSW, Clean and Invalidate of Data and Allocation Tags by Set/Way

The DC CIGDSW characteristics are:

Purpose

Clean and Invalidate data and Allocation Tags in data cache by set/way.

Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CIGDSW are UNDEFINED.

Attributes

DC CIGDSW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
SetWay																												Level			RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

DC CIGDSW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1110	0b110

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CIGDVAC, Clean and Invalidate of Data and Allocation Tags by VA to PoC

The DC CIGDVAC characteristics are:

Purpose

Clean and Invalidate data and Allocation Tags in data cache by address to Point of Coherency.

Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CIGDVAC are UNDEFINED.

Attributes

DC CIGDVAC is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
												VA Virtual address to use																				
												VA Virtual address to use																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

VA, bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CIGDVAC instruction

If EL0 access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

If EL0 access is enabled, when executed at EL0, this instruction requires read access permission to the VA, otherwise it generates a Permission fault.

For more information, see 'Permission fault'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIGDVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1110	0b101

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CIGSW, Clean and Invalidate of Allocation Tags by Set/Way

The DC CIGSW characteristics are:

Purpose

Clean and Invalidate Allocation Tags in data cache by set/way.

Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC CIGSW are UNDEFINED.

Attributes

DC CIGSW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
SetWay																													Level		RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

Executing the DC CIGSW instruction

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIGSW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1110	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.DCCISW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_SetWay);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CIGVAC, Clean and Invalidate of Allocation Tags by VA to PoC

The DC CIGVAC characteristics are:

Purpose

Clean and Invalidate Allocation Tags in data cache by address to Point of Coherency.

Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC CIGVAC are UNDEFINED.

Attributes

DC CIGVAC is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
												VA Virtual address to use																				
												VA Virtual address to use																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

VA, bits **Bits** [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CIGVAC instruction

If EL0 access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

If EL0 access is enabled, when executed at EL0, this instruction requires read access permission to the VA, otherwise it generates a Permission fault.

For more information, see 'Permission fault'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIGVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1110	0b011

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

DC CIPAE, Data or unified Cache line Clean and Invalidate by PA to PoE

The DC CIPAE characteristics are:

Purpose

Data or unified Cache line Clean and Invalidate by PA to PoE.

Configuration

This instruction is present only when FEAT_MEC is implemented. Otherwise, direct accesses to DC CIPAE are UNDEFINED.

Attributes

DC CIPAE is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Reserved, RES0.

Executing DC CIPAE

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIPAE, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoE);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoE);

```

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CIPAPA, Data or unified Cache line Clean and Invalidate by PA to PoPA

The DC CIPAPA characteristics are:

Purpose

Clean and Invalidate data cache by physical address to the Point of Physical Aliasing.

Note

This instruction cleans and invalidates all copies of the Location specified in the Xt argument, irrespective of any MECID associated with the Location. Memory accesses resulting from the Clean operation use the MECID associated with the cache entry.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to DC CIPAPA are UNDEFINED.

Attributes

DC CIPAPA is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NS	NSE	RES0										Physical address																			
Physical address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NS, bit [63]

Together with the NSE field, this field specifies the target physical address space.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

If FEAT_SEL2 is not implemented, and {NSE, NS} == {0b0, 0b0}, then no cache entries are required to be cleaned or invalidated

NSE, bit [62]

Together with the NS field, this field specifies the target physical address space.

For a description of the values derived by evaluating NS and NSE together, see DC CIPAPA.NS.

(old)

htmldiff from-

(new)

DC CISW, Data or unified Cache line Clean and Invalidate by Set/Way

The DC CISW characteristics are:

Purpose

Clean and Invalidate data cache by set/way.

Configuration

AArch64 System instruction DC CISW performs the same function as AArch32 System instruction [DCCISW](#).

Attributes

DC CISW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																																
SetWay																															Level	RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1110	0b010

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

DC CIVAC, Data or unified Cache line Clean and Invalidate by VA to PoC

The DC CIVAC characteristics are:

Purpose

Clean and Invalidate data cache by address to Point of Coherency.

Configuration

AArch64 System instruction DC CIVAC performs the same function as AArch32 System instruction [DCCIMVAC](#).

Attributes

DC CIVAC is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CIVAC instruction

If EL0 access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

If EL0 access is enabled, when executed at EL0, this instruction requires read access permission to the VA, otherwise it generates a Permission fault.

For more information, see 'Permission fault'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CIVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1110	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The DC CSW characteristics are:

Clean data cache by set/way.

AArch64 System instruction DC CSW performs the same function as AArch32 System instruction [DCCSW](#).

DC CSW is a 64-bit System instruction.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32								
																RES0																							
																								SetWay												Level		RES0	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

Reserved, RES0.

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$$A = \text{Log}_2(\text{ASSOCIATIVITY}), L = \text{Log}_2(\text{LINELEN}), B = (L + S), S = \text{Log}_2(\text{NSETS}).$$

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1010	0b010

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

DC CVAC, Data or unified Cache line Clean by VA to PoC

The DC CVAC characteristics are:

Purpose

Clean data cache by address to Point of Coherency.

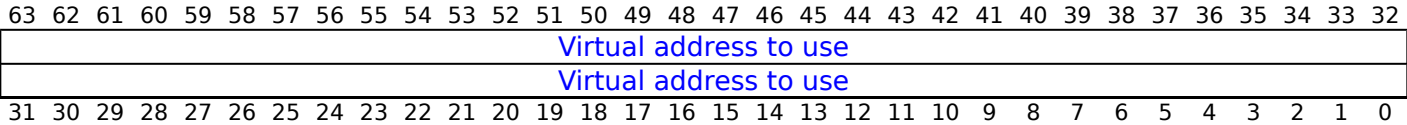
Configuration

AArch64 System instruction DC CVAC performs the same function as AArch32 System instruction [DCCMVAC](#).

Attributes

DC CVAC is a 64-bit System instruction.

Field descriptions



Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CVAC instruction

If EL0 access is enabled, when executed at EL0, ~~the this~~ instruction ~~may requires~~ ~~generates~~ ~~read access permission to the VA, otherwise it generates~~ a Permission fault, subject to the constraints described in ~~'MMU Permission faults generated by cache maintenance operations fault'~~.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1010	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

DC CVADP, Data or unified Cache line Clean by VA to PoDP

The DC CVADP characteristics are:

Purpose

Clean data cache by address to Point of Deep Persistence.
If the memory system does not identify a Point of Deep Persistence, then this instruction behaves as a [DC CVAP](#).

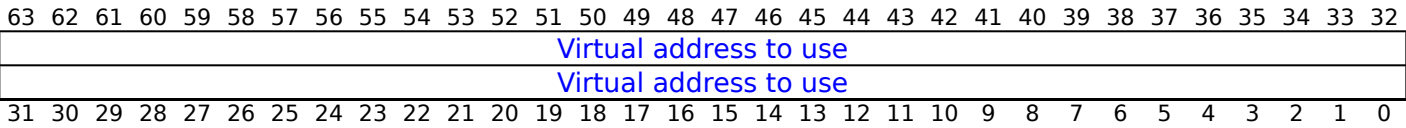
Configuration

This instruction is present only when FEAT_DPB2 is implemented. Otherwise, direct accesses to DC CVADP are UNDEFINED.

Attributes

DC CVADP is a 64-bit System instruction.

Field descriptions



Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CVADP instruction

If EL0 access is enabled, when executed at EL0, this instruction may require read access to the VA, subject to the constraints a described Permission in fault, see 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CVADP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1101	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVADP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CVAP, Data or unified Cache line Clean by VA to PoP

The DC CVAP characteristics are:

Purpose

Clean data cache by address to Point of Persistence.

If the memory system does not identify a Point of Persistence, then this instruction behaves as a [DC CVAC](#).

Configuration

This instruction is present only when FEAT_DPB is implemented. Otherwise, direct accesses to DC CVAP are UNDEFINED.

Attributes

DC CVAP is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CVAP instruction

If EL0 access is enabled, when executed at EL0, this instruction may require a read access to the VA, subject to the constraints described in [Permission faults generated by cache maintenance operations](#). If a fault occurs, see [Permission faults](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CVAP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1100	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC CVAU, Data or unified Cache line Clean by VA to PoU

The DC CVAU characteristics are:

Purpose

Clean data cache by address to Point of Unification.

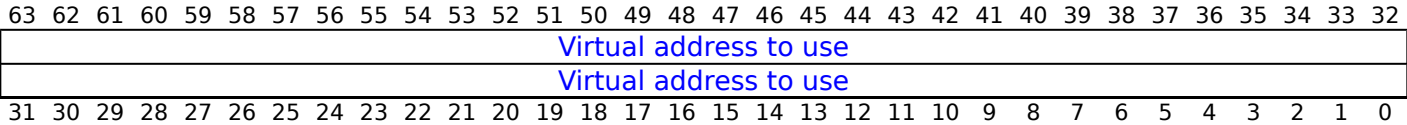
Configuration

AArch64 System instruction DC CVAU performs the same function as AArch32 System instruction [DCCMVAU](#).

Attributes

DC CVAU is a 64-bit System instruction.

Field descriptions



Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC CVAU instruction

If EL0 access is enabled, when executed at EL0, ~~the this instruction may require~~ **generates read access permission to the VA, otherwise it generates** a Permission fault, subject to the constraints described in **'MMU Permission faults generated by cache maintenance operations fault'**.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC CVAU, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1011	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TOCU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.TOCU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCCVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

DC GVA, Data Cache set Allocation Tag by VA

The DC GVA characteristics are:

Purpose

Write a value to the Allocation Tags of a naturally aligned block of N bytes, where the size of N is identified in [DCZID_EL0](#). The Allocation Tag used is determined by the input address.

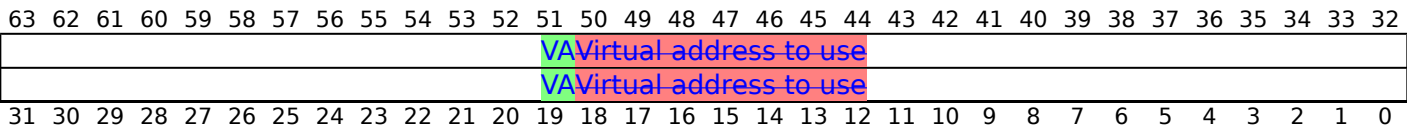
Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC GVA are UNDEFINED.

Attributes

DC GVA is a 64-bit System instruction.

Field descriptions



VA, bitsBits [63:0]

Virtual address to use. There is no alignment restriction on the address within the block of N bytes that is used.

Executing the DC GVA instruction

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is not set.

If the memory region being modified is any type of Device memory, this instruction can give an alignment fault that is prioritized in the same way as other alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of stores to each Allocation Tag within the block being accessed, and so it:

- Generates a Permission fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC GVA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0100	0b011

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.DZE == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.DZE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Tag);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Tag);
    elsif PSTATE.EL == EL2 then
        AArch64.MemZero(X[t, 64], CacheType_Tag);
    elsif PSTATE.EL == EL3 then
        AArch64.MemZero(X[t, 64], CacheType_Tag);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC GZVA, Data Cache set Allocation Tags and Zero by VA

The DC GZVA characteristics are:

Purpose

Zero data and write a value to the Allocation Tags of a naturally aligned block of N bytes, where the size of N is identified in [DCZID_ELO](#). The Allocation Tag used is determined by the input address.

Configuration

This instruction is present only when FEAT_MTE is implemented. Otherwise, direct accesses to DC GZVA are UNDEFINED.

Attributes

DC GZVA is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
												VA Virtual address to use																			
												VA Virtual address to use																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VA, bits [63:0]

Virtual address to use. There is no alignment restriction on the address within the block of N bytes that is used.

Executing the DC GZVA instruction

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is not set.

If the memory region being zeroed is any type of Device memory, this instruction can give an alignment fault which is prioritized in the same way as other alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of Stores to each byte and Allocation tag within the block being accessed, and so it:

- Generates a Permission fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC GZVA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0100	0b100

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.DZE == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.DZE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data_Tag);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data_Tag);
    elsif PSTATE.EL == EL2 then
        AArch64.MemZero(X[t, 64], CacheType_Data_Tag);
    elsif PSTATE.EL == EL3 then
        AArch64.MemZero(X[t, 64], CacheType_Data_Tag);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC IGDSW, Invalidate of Data and Allocation Tags by Set/Way

The DC IGDSW characteristics are:

Purpose

Invalidate data and Allocation Tags in data cache by set/way.

Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC IGDSW are UNDEFINED.

Attributes

DC IGDSW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
SetWay																													Level		RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b110

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

DC IGDVAC, Invalidate of Data and Allocation Tags by VA to PoC

The DC IGDVAC characteristics are:

Purpose

Invalidate data and Allocation Tags in data cache by address to Point of Coherency.

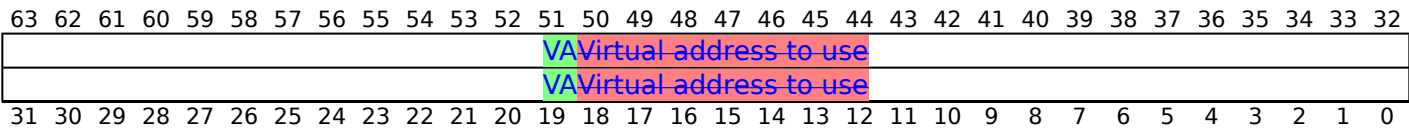
Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC IGDVAC are UNDEFINED.

Attributes

DC IGDVAC is a 64-bit System instruction.

Field descriptions



VA, bitsBits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC IGDVAC instruction

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 1.

If this EL0 instruction access requires is write enabled, access when permission executed to at the EL0VA, the otherwise instruction it may generate generates a Permission fault, subject to the constraints described in 'MMUPermission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC IGDVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TPCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCIVAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data_Tag, CacheOp_Invalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

DC IGSW, Invalidate of Allocation Tags by Set/Way

The DC IGSW characteristics are:

Purpose

Invalidate Allocation Tags in data cache by set/way.

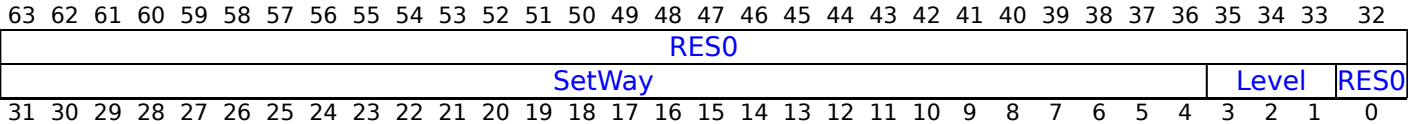
Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC IGSW are UNDEFINED.

Attributes

DC IGSW is a 64-bit System instruction.

Field descriptions



Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b100

(old)	htmldiff from-	(new)
-------	----------------	-------

DC IGVAC, Invalidate of Allocation Tags by VA to PoC

The DC IGVAC characteristics are:

Purpose

Invalidate Allocation Tags in data cache by address to Point of Coherency.

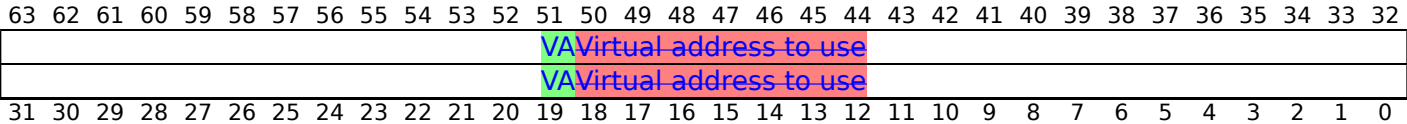
Configuration

This instruction is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to DC IGVAC are UNDEFINED.

Attributes

DC IGVAC is a 64-bit System instruction.

Field descriptions



VA, bitsBits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC IGVAC instruction

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 1.

If this EL0 instruction access requires iswrite enabled, access when permission executed to at the EL0VA, the otherwise instruction it may generate generates a Permission fault, subject to the constraints described in 'MMU Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC IGVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TPCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCIVAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Tag, CacheOp_Invalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC ISW, Data or unified Cache line Invalidate by Set/Way

The DC ISW characteristics are:

Purpose

Invalidate data cache by set/way.

When FEAT_MTE2 is implemented, this instruction might invalidate Allocation Tags from caches. When it invalidates Allocation Tags from caches, it also cleans them.

Configuration

AArch64 System instruction DC ISW performs the same function as AArch32 System instruction [DCISW](#).

Attributes

DC ISW is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																																	
SetWay																															Level		RES0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:32]

Reserved, RES0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, RES0.

Executing the DC ISW instruction

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC ISW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.DCISW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_SetWay);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC IVAC, Data or unified Cache line Invalidate by VA to PoC

The DC IVAC characteristics are:

Purpose

Invalidate data cache by address to Point of Coherency.

When FEAT_MTE2 is implemented, this instruction might invalidate Allocation Tags from caches. When it invalidates Allocation Tags from caches, it also cleans them.

Configuration

AArch64 System instruction DC IVAC performs the same function as AArch32 System instruction [DCIMVAC](#).

Attributes

DC IVAC is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DC IVAC instruction

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 1.

If this EL0 instruction access requires is write enabled, access when permission executed to at the EL0 VA, the otherwise instruction it may generate generates a Permission fault, subject to the constraints described in 'MMU Permission faults generated by cache maintenance operations fault'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC IVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TPCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCIVAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_PoC);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DC ZVA, Data Cache Zero by VA

The DC ZVA characteristics are:

Purpose

Zero data cache by address. Zeroes a naturally aligned block of N bytes, where the size of N is identified in [DCZID_EL0](#).

Configuration

There are no configuration notes.

Attributes

DC ZVA is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Virtual address to use. There is no alignment restriction on the address within the block of N bytes that is used.

Executing the DC ZVA instruction

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 0.

If the memory region being zeroed is any type of Device memory, this instruction can give an Alignment fault which is prioritized in the same way as other Alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of Stores to each byte within the block being accessed, and so it:

- Generates a Permission fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

Accesses to this instruction use the following encodings in the System instruction encoding space:

DC ZVA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0100	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLRL_EL1.DZE == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLRL_EL2.DZE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DCZVA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data);
    elsif PSTATE.EL == EL2 then
        AArch64.MemZero(X[t, 64], CacheType_Data);
    elsif PSTATE.EL == EL3 then
        AArch64.MemZero(X[t, 64], CacheType_Data);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MRS <Xt>, DCZID_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b111

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DCZID_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DCZID_EL0;
elseif PSTATE.EL == EL2 then
    X[t, 64] = DCZID_EL0;
elseif PSTATE.EL == EL3 then
    X[t, 64] = DCZID_EL0;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DSPSR_EL0, Debug Saved Program Status Register

The DSPSR_EL0 characteristics are:

Purpose

Holds the saved process state for Debug state. On entering Debug state, PSTATE information is written to this register. On exiting Debug state, values are copied from this register to PSTATE.

Configuration

AArch64 System register DSPSR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [DSPSR\[31:0\]](#).

Attributes

DSPSR_EL0 is a 64-bit register.

Field descriptions

When AArch32 is supported and exiting Debug state to AArch32 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
N	Z	C	V	Q	IT[1:0]	DIT	SSBS	PAN	SS	IL	GE				IT[7:2]				E	A	I	F	T	M[4]		M[3:0]					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

N, bit [31]

Negative Condition flag. Copied to PSTATE.N on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Copied to PSTATE.Z on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Copied to PSTATE.C on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Copied to PSTATE.V on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Q, bit [27]

Overflow or saturation flag. Copied to PSTATE.Q on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IT, bits [15:10, 26:25]

If-Then. Copied to PSTATE.IT on exiting Debug state.

DSPSR_EL0.IT must contain a value that is valid for the instruction being returned to.

The IT field is split as follows:

- IT[1:0] is DSPSR_EL0[26:25].
- IT[7:2] is DSPSR_EL0[15:10].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIT, bit [24]**When FEAT_DIT is implemented:**

Data Independent Timing. Copied to PSTATE.DIT on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [23]**When FEAT_SSBS is implemented:**

Speculative Store Bypass. Copied to PSTATE.SSBS on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Copied to PSTATE.PAN on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Copied to PSTATE.SS on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Copied to PSTATE.IL on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GE, bits [19:16]

Greater than or Equal flags. Copied to PSTATE.GE on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E, bit [9]

Endianness. Copied to PSTATE.E on exiting Debug state.

If the implementation does not support big-endian operation, DPSR_EL0.E is RES0. If the implementation does not support little-endian operation, DPSR_EL0.E is RES1. On exiting Debug state, if the implementation does not support big-endian operation at the Exception level being returned to, DPSR_EL0.E is RES0, and if the implementation does not support little-endian operation at the Exception level being returned to, DPSR_EL0.E is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Copied to PSTATE.A on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Copied to PSTATE.I on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Copied to PSTATE.F on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T, bit [5]

T32 Instruction set state. Copied to PSTATE.T on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[4], bit [4]

Execution state. Copied to PSTATE.nRW on exiting Debug state.

M[4]	Meaning
0b1	AArch32 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch32 Mode. Copied to PSTATE.M[3:0] on exiting Debug state.

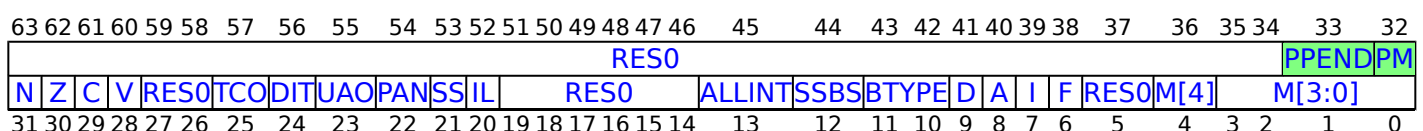
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0110	Monitor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

Other values are reserved. If DPSR_EL0.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, exiting Debug state is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When AArch64 is supported and entering or exiting Debug state from or to AArch64 state:



Bits [63:34:32]

Reserved, RES0.

PPEND, bit [33]**When FEAT_SEBEP is implemented:**

PMU exception pending bit. Set to the value of PSTATE.PPEND on entering Debug state, and conditionally copied to PSTATE.PPEND on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PM, bit [32]**When FEAT_EBEP is implemented:**

PMU exception mask bit. Set to the value of PSTATE.PM on entering Debug state, and copied to PSTATE.PM on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on entering Debug state, and copied to PSTATE.N on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on entering Debug state, and copied to PSTATE.Z on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on entering Debug state, and copied to PSTATE.C on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on entering Debug state, and copied to PSTATE.V on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

TCO, bit [25]

When FEAT_MTE is implemented:

Tag Check Override. Set to the value of PSTATE.TCO on entering Debug state, and copied to PSTATE.TCO on exiting Debug state.

When FEAT_MTE2 is not implemented, it is CONSTRAINED UNPREDICTABLE whether this field is RES0 or behaves as if FEAT_MTE2 is implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DIT, bit [24]

When FEAT_DIT is implemented:

Data Independent Timing. Set to the value of PSTATE.DIT on entering Debug state, and copied to PSTATE.DIT on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UAO, bit [23]

When FEAT_UAO is implemented:

User Access Override. Set to the value of PSTATE.UAO on entering Debug state, and copied to PSTATE.UAO on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]

When FEAT_PAN is implemented:

Privileged Access Never. Set to the value of PSTATE.PAN on entering Debug state, and copied to PSTATE.PAN on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on entering Debug state, and conditionally copied to PSTATE.SS on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on entering Debug state, and copied to PSTATE.IL on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:14]

Reserved, RES0.

ALLINT, bit [13]

When FEAT_NMI is implemented:

All IRQ or FIQ interrupts mask. Set to the value of PSTATE.ALLINT on entering Debug state, and copied to PSTATE.ALLINT on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [12]

When FEAT_SSBS is implemented:

Speculative Store Bypass. Set to the value of PSTATE.SSBS on entering Debug state, and copied to PSTATE.SSBS on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BTYP, bits [11:10]**When FEAT_BTI is implemented:**

Branch Type Indicator. Set to the value of PSTATE.BTYPE on entering Debug state, and copied to PSTATE.BTYPE on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D, bit [9]

Debug exception mask. Set to the value of PSTATE.D on entering Debug state, and copied to PSTATE.D on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Set to the value of PSTATE.A on entering Debug state, and copied to PSTATE.A on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on entering Debug state, and copied to PSTATE.I on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on entering Debug state, and copied to PSTATE.F on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

M[4], bit [4]

Execution state. Set to 0b0, the value of PSTATE.nRW, on entering Debug state from AArch64 state, and copied to PSTATE.nRW on exiting Debug state.

M[4]	Meaning
0b0	AArch64 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch64 Exception level and selected Stack Pointer.

M[3:0]	Meaning
0b0000	EL0t.
0b0100	EL1t.
0b0101	EL1h.
0b1000	EL2t.
0b1001	EL2h.
0b1100	EL3t.
0b1101	EL3h.

Other values are reserved. If DPSR_EL0.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, exiting Debug state is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The bits in this field are interpreted as follows:

- M[3:2] is set to the value of PSTATE.EL on entering Debug state and copied to PSTATE.EL on exiting Debug state.
- M[1] is unused and is 0 for all non-reserved values.
- M[0] is set to the value of PSTATE.SP on entering Debug state and copied to PSTATE.SP on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing DPSR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, DPSR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

```
if !Halted() then
    UNDEFINED;
else
    X[t, 64] = DPSR_EL0;
```

MSR DPSR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

```
if !Halted() then
    UNDEFINED;
else
    DPSR_EL0 = X[t, 64];
```

(old)**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

DVP RCTX, Data Value Prediction Restriction by Context

The DVP RCTX characteristics are:

Purpose

Data Value Prediction Restriction by Context applies to all Data Value Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Data value predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when FEAT_SPECRES is implemented. Otherwise, direct accesses to DVP RCTX are UNDEFINED.

Attributes

DVP RCTX is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0															GVMID	VMID															
RES0				NSEN	NS	EL	RES0								GASID	ASID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:49]

Reserved, RES0.

GVMID, bit [48]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, then this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

VMID, bits [47:32]

Only applies when bit[48] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)).

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

If the implementation supports 16 bits of VMID, then the upper 8 bits of the VMID must be written to 0 by software when the context being affected only uses 8 bits.

Bits [31:28]

Reserved, RES0.

NSE, bit [27]

When FEAT_RME is implemented:

Together with the NS field, selects the Security state.

For a description of the values derived by evaluating NS and NSE together, see DVP_RCTX.NS.

Otherwise:

Reserved, RES0.

NS, bit [26]

When FEAT_RME is implemented:

Together with the NSE field, selects the Security state. Defined values are:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Some Effective values are determined by the current Security state:

- When executed in Secure state, the Effective value of NSE is 0.
- When executed in Non-secure state, the Effective value of {NSE, NS} is {0, 1}.

- When executed in Realm state, the Effective value of {NSE, NS} is {1, 1}.

This instruction is treated as a NOP when executed at EL3 and either:

An instruction with an EL field that has a value other than 0b11 (EL3) is treated as a NOP when executed at EL3 with DVP_RCTX.{NSE, NS} == {1, 0}.

- DVP_RCTX.{NSE, NS} selects a reserved value.
- DVP_RCTX.{NSE, NS} == {1, 0} and DVP_RCTX.EL has a value other than 0b11.

Otherwise:

Security State. Defined values are:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

When executed in Non-secure state, the Effective value of NS is 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

Bits [23:17]

Reserved, RES0.

GASID, bit [16]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [15:0]

Only applies for an EL0 target execution context and when bit[16] is 0.

Otherwise this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.

Executing the DVP RCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

DVP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b101

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnRCTX == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DVPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnRCTX == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.DVPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERRIDR_EL1, Error Record ID Register

The ERRIDR_EL1 characteristics are:

Purpose

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configuration

AArch64 System register ERRIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERRIDR\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERRIDR_EL1 are UNDEFINED.

Attributes

ERRIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																NUM															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

NUM, bits [15:0]

Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers.

Each implemented record is owned by a node. A node might own multiple records.

Accessing ERRIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERRIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERRIDR_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERRIDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ERRIDR_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERRSELR_EL1, Error Record Select Register

The ERRSELR_EL1 characteristics are:

Purpose

Selects an error record to be accessed through the Error Record System registers.

Configuration

AArch64 System register ERRSELR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERRSELR\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERRSELR_EL1 are UNDEFINED.

If [ERRIDR_EL1](#) indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

ERRSELR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																SEL															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

SEL, bits [15:0]

Selects the error record accessed through the ERX registers.

For example, if ERRSELR_EL1.SEL is 0x0004, then direct reads and writes of [ERXSTATUS_EL1](#) access ERR4STATUS.

If ERRSELR_EL1.SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then all of the following apply:

- The value read back from ERRSELR_EL1.SEL is UNKNOWN.
- One of the following occurs:
 - An UNKNOWN error record is selected.
 - The ERX*_EL1 registers are RAZ/WI.
 - ERX*_EL1 register reads and writes are NOPS.
 - ERX*_EL1 register reads and writes are UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ERRSELR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERRSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERRSELR_EL1;

```

MSR ERRSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERRSELR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERRSELR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERRSELR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXADDR_EL1, Selected Error Record Address Register

The ERXADDR_EL1 characteristics are:

Purpose

Accesses [ERR<n>ADDR](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

AArch64 System register ERXADDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXADDR\[31:0\]](#).

AArch64 System register ERXADDR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXADDR2\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXADDR_EL1 are UNDEFINED.

Attributes

ERXADDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
														ERR<n>ADDR																		
														ERR<n>ADDR																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:0]

ERXADDR_EL1 accesses [ERR<n>ADDR](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXADDR_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXADDR_EL1 is RAZ/WI.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are UNDEFINED.

[ERR<n>ADDR](#) describes additional constraints that also apply when [ERR<n>ADDR](#) is accessed through ERXADDR_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXADDR_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b000	0b0101	0b0100	0b011
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXADDR_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERXADDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ERXADDR_EL1;
    
```

MSR ERXADDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXADDR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXADDR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXADDR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXCTLR_EL1, Selected Error Record Control Register

The ERXCTLR_EL1 characteristics are:

Purpose

Accesses [ERR<n>CTLR](#) for the error record <n> selected by [ERRSELR_EL1.SEL](#).

Configuration

AArch64 System register ERXCTLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXCTLR\[31:0\]](#).

AArch64 System register ERXCTLR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXCTLR2\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXCTLR_EL1 are UNDEFINED.

Attributes

ERXCTLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>CTLR																															
ERR<n>CTLR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXCTLR_EL1 accesses [ERR<n>CTLR](#), where <n> is the value in [ERRSELR_EL1.SEL](#).

Accessing ERXCTLR_EL1

If [ERRIDR_EL1.NUM](#) is 0x0000 or [ERRSELR_EL1.SEL](#) is greater than or equal to [ERRIDR_EL1.NUM](#), then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXCTLR_EL1 is RAZ/WI.
- Direct reads and writes of ERXCTLR_EL1 are NOPs.
- Direct reads and writes of ERXCTLR_EL1 are UNDEFINED.

If [ERRSELR_EL1.SEL](#) is not the index of the first error record owned by a node, then [ERR<n>CTLR](#) is not present, meaning reads and writes of ERXCTLR_EL1 are RES0.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXCTLR_EL1;

```

MSR ERXCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXCTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXCTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXCTLR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXFR_EL1, Selected Error Record Feature Register

The ERXFR_EL1 characteristics are:

Purpose

Accesses [ERR<n>FR](#) for the error record <n> selected by [ERRSELR_EL1.SEL](#).

Configuration

AArch64 System register ERXFR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXFR\[31:0\]](#).

AArch64 System register ERXFR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXFR2\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXFR_EL1 are UNDEFINED.

Attributes

ERXFR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>FR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXFR_EL1 accesses [ERR<n>FR](#), where <n> is the value in [ERRSELR_EL1.SEL](#).

Accessing ERXFR_EL1

If [ERRIDR_EL1.NUM](#) is 0x0000 or [ERRSELR_EL1.SEL](#) is greater than or equal to [ERRIDR_EL1.NUM](#), then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXFR_EL1 is RAZ.
- Direct reads of ERXFR_EL1 are NOPs.
- Direct reads of ERXFR_EL1 are UNDEFINED.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXFR_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0

The ERXMISC0_EL1 characteristics are:

Purpose

Accesses [ERR<n>MISC0](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

AArch64 System register ERXMISC0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXMISC0\[31:0\]](#).

AArch64 System register ERXMISC0_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXMISC1\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXMISC0_EL1 are UNDEFINED.

Attributes

ERXMISC0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ERR<n>MISC0															
																ERR<n>MISC0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXMISC0_EL1 accesses [ERR<n>MISC0](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXMISC0_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC0_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are UNDEFINED.

[ERR<n>MISC0](#) describes additional constraints that also apply when [ERR<n>MISC0](#) is accessed through ERXMISC0_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXMISC0_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b000	0b0101	0b0101	0b000
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXMISC0_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERXMISC0_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ERXMISC0_EL1;
    
```

MSR ERXMISC0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXMISC0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXMISC0_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

The ERXMISC1_EL1 characteristics are:

Purpose

Accesses [ERR<n>MISC1](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

AArch64 System register ERXMISC1_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXMISC2\[31:0\]](#).

AArch64 System register ERXMISC1_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXMISC3\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXMISC1_EL1 are UNDEFINED.

Attributes

ERXMISC1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ERR<n>MISC1															
																ERR<n>MISC1															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXMISC1_EL1 accesses [ERR<n>MISC1](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXMISC1_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC1_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are UNDEFINED.

[ERR<n>MISC1](#) describes additional constraints that also apply when [ERR<n>MISC1](#) is accessed through ERXMISC1_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXMISC1_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b000	0b0101	0b0101	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXMISC1_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERXMISC1_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ERXMISC1_EL1;
    
```

MSR ERXMISC1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXMISC1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXMISC1_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

The ERXMISC2_EL1 characteristics are:

Purpose

Accesses [ERR<n>MISC2](#) for the error record <n> selected by [ERRSEL_R_EL1](#).SEL.

Configuration

AArch64 System register ERXMISC2_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXMISC4\[31:0\]](#).

AArch64 System register ERXMISC2_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXMISC5\[31:0\]](#).

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXMISC2_EL1 are UNDEFINED.

Attributes

ERXMISC2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ERR<n>MISC2															
																ERR<n>MISC2															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXMISC2_EL1 accesses [ERR<n>MISC2](#), where <n> is the value in [ERRSEL_R_EL1](#).SEL.

Accessing ERXMISC2_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSEL_R_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC2_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC2_EL1 are NOPs.
- Direct reads and writes of ERXMISC2_EL1 are UNDEFINED.

[ERR<n>MISC2](#) describes additional constraints that also apply when [ERR<n>MISC2](#) is accessed through ERXMISC2_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXMISC2_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b000	0b0101	0b0101	0b010
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXMISC2_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERXMISC2_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ERXMISC2_EL1;
    
```

MSR ERXMISC2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXMISC2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXMISC2_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

The ERXMISC3_EL1 characteristics are:

Purpose

Accesses [ERR<n>MISC3](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

AArch64 System register ERXMISC3_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXMISC6\[31:0\]](#).

AArch64 System register ERXMISC3_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ERXMISC7\[31:0\]](#).

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXMISC3_EL1 are UNDEFINED.

Attributes

ERXMISC3_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ERR<n>MISC3															
																ERR<n>MISC3															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXMISC3_EL1 accesses [ERR<n>MISC3](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXMISC3_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC3_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are UNDEFINED.

[ERR<n>MISC3](#) describes additional constraints that also apply when [ERR<n>MISC3](#) is accessed through ERXMISC3_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXMISC3_EL1

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b000	0b0101	0b0101	0b011
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC3_EL1;
    
```

MSR ERXMISC3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXMISCn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXMISC3_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXMISC3_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXMISC3_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register

The ERXPFGCDN_EL1 characteristics are:

Purpose

Accesses [ERR<n>PFGCDN](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXPFGCDN_EL1 are UNDEFINED.

Attributes

ERXPFGCDN_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>PFGCDN																															
ERR<n>PFGCDN																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXPFGCDN_EL1 accesses [ERR<n>PFGCDN](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXPFGCDN_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

If [ERRSELR_EL1](#).SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

Note

A node does not implement the Common Fault Injection Model Extension if [ERR<q>FR](#).INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in [ERRSELR_EL1](#).SEL. If the node owns a single record, then q = n.

If [ERRSEL_EL1.SEL](#) is not the index of the first error record owned by a node, then [ERR<n>PFGCDN](#) is not present, meaning reads and writes of ERXPFPGCDN_EL1 are RES0.

[ERR<n>PFGCDN](#) describes additional constraints that also apply when [ERR<n>PFGCDN](#) is accessed through ERXPFPGCDN_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXPFPGCDN_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXPFPGCDN_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXPFPGCDN_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXPFPGCDN_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFPGCDN_EL1;
    
```

MSR ERXPFPGCDN_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXPFPGCDN_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXPFPGCDN_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXPFPGCDN_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXPFPGCDN_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register

The ERXPFGCTL_EL1 characteristics are:

Purpose

Accesses [ERR<n>PFGCTL](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXPFGCTL_EL1 are UNDEFINED.

Attributes

ERXPFGCTL_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>PFGCTL																															
ERR<n>PFGCTL																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXPFGCTL_EL1 accesses [ERR<n>PFGCTL](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXPFGCTL_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

If [ERRSELR_EL1](#).SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

Note

A node does not implement the Common Fault Injection Model Extension if [ERR<q>FR](#).INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in [ERRSELR_EL1](#).SEL. If the node owns a single record, then q = n.

If [ERRSEL_EL1.SEL](#) is not the index of the first error record owned by a node, then [ERR<n>PFGCTL](#) is not present, meaning reads and writes of ERXPFPGCTL_EL1 are RES0.

[ERR<n>PFGCTL](#) describes additional constraints that also apply when [ERR<n>PFGCTL](#) is accessed through ERXPFPGCTL_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXPFPGCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXPFPGCTL_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFPGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFPGCTL_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFPGCTL_EL1;

```

MSR ERXPFPGCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXPFPGCTL_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXPFPGCTL_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXPFPGCTL_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXPFPGCTL_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register

The ERXPFGF_EL1 characteristics are:

Purpose

Accesses [ERR<n>PFGF](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXPFGF_EL1 are UNDEFINED.

Attributes

ERXPFGF_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>PFGF																															
ERR<n>PFGF																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXPFGF_EL1 accesses [ERR<n>PFGF](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXPFGF_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

If [ERRSELR_EL1](#).SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

Note

A node does not implement the Common Fault Injection Model Extension if [ERR<q>FR](#).INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in [ERRSELR_EL1](#).SEL. If the node owns a single record, then q = n.

If [ERRSEL_EL1.SEL](#) is not the index of the first error record owned by a node, then [ERR<n>PFGF](#) is not present, meaning reads of [ERXPFGF_EL1](#) are RES0.

[ERR<n>PFGF](#) describes additional constraints that also apply when [ERR<n>PFGF](#) is accessed through [ERXPFGF_EL1](#).

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXPFGF_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
& HFGTR_EL2.ERXPFGF_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGF_EL1;
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ERXSTATUS_EL1, Selected Error Record Primary Status Register

The ERXSTATUS_EL1 characteristics are:

Purpose

Accesses [ERR<n>STATUS](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

AArch64 System register ERXSTATUS_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ERXSTATUS\[31:0\]](#).

This register is present only when FEAT_RAS is implemented. Otherwise, direct accesses to ERXSTATUS_EL1 are UNDEFINED.

Attributes

ERXSTATUS_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ERR<n>STATUS																															
ERR<n>STATUS																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

ERXSTATUS_EL1 accesses [ERR<n>STATUS](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXSTATUS_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXSTATUS_EL1 is RAZ/WI.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are UNDEFINED.

[ERR<n>STATUS](#) describes additional constraints that also apply when [ERR<n>STATUS](#) is accessed through ERXSTATUS_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ERXSTATUS_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ERXSTATUS_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXSTATUS_EL1;
    
```

MSR ERXSTATUS_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ERXSTATUS_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ERXSTATUS_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TERR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TERR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ERXSTATUS_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        ERXSTATUS_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ESR_EL1, Exception Syndrome Register (EL1)

The ESR_EL1 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL1.

Configuration

AArch64 System register ESR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DFSR\[31:0\]](#).

Attributes

ESR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								ISS2																							
EC						IL	ISS																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ESR_EL1 is made UNKNOWN as a result of an exception return from EL1.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL1, the value of ESR_EL1 is UNKNOWN. The value written to ESR_EL1 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:5637]

Reserved, RES0.

Otherwise:

Reserved, RES0.

ISS2, bits [55:32]

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

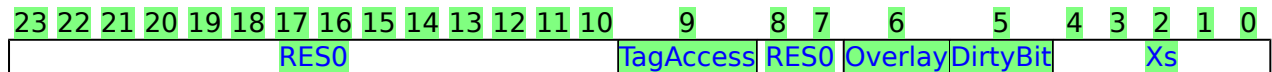
ISS2 encoding for an exception, the bit assignments are:

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

ISS2 encoding for an exception from a Data Abort (EC == 0b100100 or EC == 0b100101)



Bits [23:10]

Reserved, RES0.

TagAccess, bit [9]

When FEAT_MTE_PERM is implemented:

NoTagAccess fault.

If a memory access generates a Data Abort for a Permission fault, this field indicates whether the fault is due to the NoTagAccess memory attribute.

TagAccess	Meaning
0b0	Permission fault is not due to the NoTagAccess memory attribute.
0b1	Permission fault is due to the NoTagAccess memory attribute.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [8:7]

Reserved, RES0.

Overlay, bit [6]

When FEAT_S1POE is implemented:

Overlay flag.

If a memory access generates a Data Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Data Abort is not due to Overlay Permissions.
0b1	Data Abort due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DirtyBit, bit [5]**When FEAT_S1PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, then this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to dirty state.
0b1	Permission Fault is due to dirty state.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Xs, bits [4:0]**When FEAT_LS64 is implemented:**

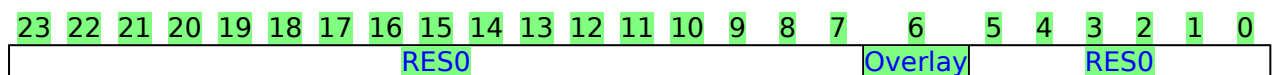
When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

Otherwise:

Reserved, RES0.

ISS2 encoding for an exception from an Instruction Abort (EC == 0b100000 or EC == 0b100001)**Bits [23:7]**

Reserved, RES0.

Overlay, bit [6]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag.

If a memory access generates a Instruction Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Instruction Abort is not due to Overlay Permissions.
0b1	Instruction Abort due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:0]

Reserved, RES0.

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

EC	Meaning	ISS	Applies when
0b000000	Unknown reason.	ISS encoding for exceptions with an unknown reason	
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS encoding for an exception from a WF* instruction	
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCRR or MRRC access	When AArch32 is supported
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: <ul style="list-style-type: none"> An STC to write data to memory from DBGDTRRXint. An LDC to read data from memory to DBGDTRTXint. 	ISS encoding for an exception from an LDC or STC instruction	When AArch32 is supported
0b000111	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN , CPTR_EL2.FPEN , CPTR_EL2.TFP , or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.	ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality resulting from the FPEN and TFP traps	
0b001010	Trapped execution of an LD64B or ST64B* instruction.	ISS encoding for an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS encoding for an exception	When AArch32 is supported

0b001101	Branch Target Exception.	from an MCRR or MRRC access ISS encoding for an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
0b001110	Illegal Execution state.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b010001	SVC instruction execution in AArch32 state.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch32 is supported
0b010101	SVC instruction execution in AArch64 state.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch64 is supported
0b011000	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001, or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
0b010100	Trapped MSRR, MRRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000.	ISS encoding for an exception from MSRR, MRRS, or 128-bit System instruction execution in AArch64 state	When FEAT_SYSREG128 is implemented or FEAT_SYSINSTR128 is implemented
0b011001	Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.	ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented
0b011011	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when	ISS encoding for an exception from a TSTART instruction	When FEAT_TME is implemented

	SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.		
0b011100	Exception from a Pointer Authentication instruction authentication failure	ISS encoding for an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
0b011101	Access to SME functionality trapped as a result of CPACR_EL1.SMEN , CPTR_EL2.SMEN , CPTR_EL2.TSM , CPTR_EL3.ESM , or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.	ISS encoding for an exception due to SME functionality	When FEAT_SME is implemented
0b011110	Exception from a Granule Protection Check	ISS encoding for an exception from a Granule Protection Check	When FEAT_RME is implemented
0b100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from an Instruction Abort	
0b100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from an Instruction Abort	
0b100010	PC alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	

0b100100	Data Abort exception from a lower Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from a Data Abort	
0b100101	Data Abort exception taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from a Data Abort	
0b100110	SP alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100111	Memory Operation Exception.	ISS encoding for an exception from the Memory Copy and Memory Set instructions	When FEAT_MOPS is implemented
0b101000	Trapped floating-point exception taken from AArch32 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS encoding for an exception from a trapped floating-point exception	When AArch32 is supported

0b101100	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS encoding for an exception from a trapped floating-point exception	When AArch64 is supported
0b101111	SError interrupt.	ISS encoding for an SError interrupt	
0b110000	Breakpoint exception from a lower Exception level.	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception	
0b110001	Breakpoint exception taken without a change in Exception level.	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception	
0b110010	Software Step exception from a lower Exception level.	ISS encoding for an exception from a Software Step exception	
0b110011	Software Step exception taken without a change in Exception level.	ISS encoding for an exception from a Software Step exception	
0b110100	Watchpoint exception from a lower Exception level.	ISS encoding for an exception from a Watchpoint exception	
0b110101	Watchpoint exception taken without a change in Exception level.	ISS encoding for an exception from a Watchpoint exception	
0b111000	BKPT instruction execution in AArch32 state.	ISS encoding for an exception from execution of a Breakpoint instruction	When AArch32 is supported
0b111100	BRK instruction execution in AArch64 state.	ISS encoding for an exception from execution of a Breakpoint instruction	When AArch64 is supported
0b111101	PMU exception	ISS encoding for a PMU exception	When FEAT_EBEP is implemented

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.

- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is **CONSTRAINED UNPREDICTABLE**.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally **UNKNOWN** value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

IL	Meaning
0b0	16-bit instruction trapped.
0b1	32-bit instruction trapped. This value is also used when the exception is one of the following: <ul style="list-style-type: none"> An SError interrupt. An Instruction Abort exception. A PC alignment fault exception. An SP alignment fault exception. A Data Abort exception for which the value of the ISV bit is 0. An Illegal Execution state exception. Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: <ul style="list-style-type: none"> 0b0: 16-bit T32 BKPT instruction. 0b1: 32-bit A32 BKPT instruction or A64 BRK instruction. An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally **UNKNOWN** value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not **UNPREDICTABLE**, the field takes the value 0b11111.
- If the instruction that generated the exception was **UNPREDICTABLE**, the field takes an **UNKNOWN** value that must be either:
 - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
 - The value 0b11111.

ISS encoding for exceptions with an unknown reason

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								

Bits [24:0]

Reserved, RES0.

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
 - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
 - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
 - Instruction encodings that are unallocated.
 - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the [SCTLR_EL1](#).{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
 - An HVC instruction when disabled by [HCR_EL2](#).HCD or [SCR_EL3](#).HCE.
 - An SMC instruction when disabled by [SCR_EL3](#).SMD.
 - An HLT instruction when disabled by [EDSCR](#).HDE.
- Attempted execution of an MSR or MRS instruction to access [SP_EL0](#) when the value of [SPSel](#).SP is 0.
- Attempted execution of an MSR or MRS instruction using a [_EL12](#) register name when [HCR_EL2](#).E2H == 0.
- Attempted execution, in Debug state, of:
 - A DCPS1 instruction when the value of [HCR_EL2](#).TGE is 1 and EL2 is disabled or not implemented in the current Security state.
 - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
 - A DCPS3 instruction when the value of [EDSCR](#).SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using [R13_mon](#).
- In Debug state when the value of [EDSCR](#).SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to [SPSR_mon](#), [SP_mon](#), or [LR_mon](#).
- An exception that is taken to EL2 because the value of [HCR_EL2](#).TGE is 1 that, if the value of [HCR_EL2](#).TGE was 0 would have been reported with an [ESR_ELx](#).EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF* instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0										RN			RES0		RV	TI			

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]

When FEAT_WFxT is implemented:

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [4:3]

Reserved, RES0.

RV, bit [2]**When FEAT_WFxT is implemented:**

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

RV	Meaning
0b0	Register field invalid.
0b1	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

TI	Meaning	Applies when
0b00	WFI trapped.	
0b01	WFE trapped.	
0b10	WFIT trapped.	When FEAT_WFxT is implemented
0b11	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating this exception:

- [SCTLR_EL1](#).{nTWE, nTWI}.
- [HCR_EL2](#).{TWE, TWI}.
- [SCR_EL3](#).{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc2		Opc1		CRn				Rt				CRm				Direction			

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCR instruction.
0b1	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.

- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#) or [CPACR](#) using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CNTHCTL_EL2](#).EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL3](#).TCPAC, for accesses to [CPACR](#) from EL1 and EL2, and accesses to [HCPTR](#) from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

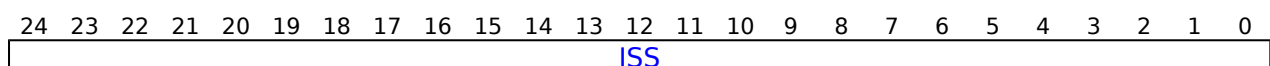
The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- [CPACR_EL1](#).TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [HCR_EL2](#).TID0, for accesses to the [JIDR](#) register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- [MDCR_EL3](#).TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- [HCR_EL2](#).TID0, for accesses to the [FPSID](#) register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- [HCR_EL2](#).TID3, for accesses to registers in ID group 3 including [MVFR0](#), [MVFR1](#) and [MVFR2](#), VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B* instruction



ISS, bits [24:0]

ISS	Meaning	Applies when
0b000000000000000000000000	ST64BV instruction trapped.	When FEAT_LS64_V is implemented
0b000000000000000000000001	ST64BV0 instruction trapped.	When FEAT_LS64_ACCDATA is implemented
0b000000000000000000000010	LD64B or ST64B instruction trapped.	When FEAT_LS64 is implemented

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc1				RES0	Rt2				Rt				CRm				Direction		

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these

definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCRR instruction.
0b1	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, [HDFGRTR_EL2](#).PMCCNTR_EL0 for MRRC access and [HDFGWTR_EL2](#).PMCCNTR_EL0 for MCRR access to [PMCCNTR](#) at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- [MDSCR_EL1](#).TDCC, for accesses to the Debug ROM registers [DBGDSAR](#) and [DBGDRAR](#) at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- [CPACR_EL1](#).TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

Note

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				imm8								RES0	Rn				Offset	AM		Direction			

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Offset	Meaning
0b0	Subtract offset.
0b1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
0b000	Immediate unindexed.
0b001	Immediate post-indexed.
0b010	Immediate offset.
0b011	Immediate pre-indexed.
0b100	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to memory. STC instruction.
0b1	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- [MDSCR_EL1](#).TDCC, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPE and TFP traps

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0																			

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

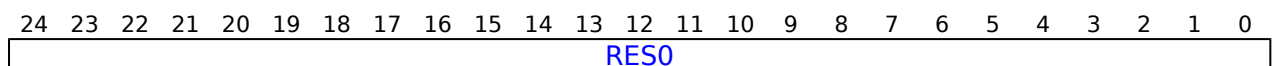
Bits [19:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- [CPACR_EL1.FPEN](#), for accesses to SIMD and floating-point registers trapped to EL1.
- [CPTR_EL2.FPEN](#) and [CPTR_EL2.TFP](#), for accesses to SIMD and floating-point registers trapped to EL2.
- [CPTR_EL3.TFP](#), for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTTR_EL2.ZEN, CPTTR_EL2.TZ, or CPTTR_EL3.EZ



The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

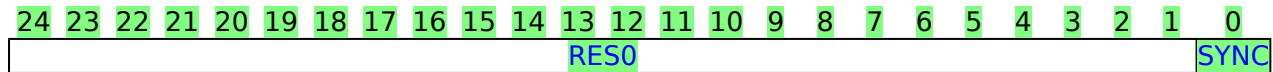
Bits [24:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- [CPACR_EL1.ZEN](#), for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- [CPTR_EL2.ZEN](#) and [CPTR_EL2.TZ](#), for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- [CPTR_EL3.EZ](#), for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for a PMU exception



Bits [24:1]

Reserved, RES0.

SYNC, bit [0]

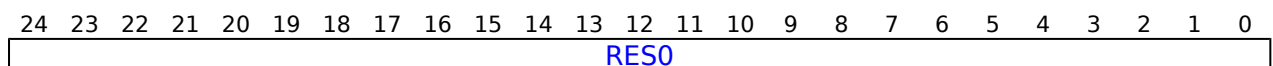
Indicates whether the exception was taken synchronously or asynchronously.

SYNC	Meaning	Applies when
0b0	The exception was taken asynchronously because an overflow status flag was set.	
0b1	The exception was taken synchronously because PSTATE.PPEND was set.	When FEAT_SEBEP is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault



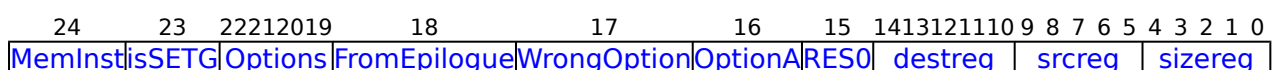
Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from the Memory Copy and Memory Set instructions



MemInst, bit [24]

Indicates the memory instruction class causing the exception.

MemInst	Meaning
0b0	CPYFE*, CPYFM*, CPYE*, and CPYM* instructions.
0b1	SETE*, SETM*, SETGE*, and SETGM* instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

isSETG, bit [23]

Indicates whether the instruction belongs to SETGM* or SETGE* class of instruction.

isSETG	Meaning
0b0	Not a SETGM* or SETGE* instruction.
0b1	SETGM* or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Options, bits [22:19]

Options : the Options field of the instruction.

For Memory Copy instructions, bits[22:19] forms the Options field, which holds the bits[15:12] of the instruction.

For Memory Set instructions:

- Bits[22:21] are RES0.
- Bits[20:19] form the Options field, which holds the bits[13:12] of the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FromEpilogue, bit [18]

Indicates whether the instruction belongs to the epilogue class of Memory Copy or Memory Set instructions.

FromEpilogue	Meaning
0b0	Not an epilogue instruction.
0b1	CPYE*, CPYFE*, SETE*, or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WrongOption, bit [17]

Algorithm option.

WrongOption	Meaning
0b0	WrongOption is false.
0b1	WrongOption is true.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OptionA, bit [16]

Algorithm type indicated by the PSTATE.C bit.

OptionA	Meaning
0b0	OptionB indicated by PSTATE.C is 0.
0b1	OptionA indicated by PSTATE.C is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

destreg, bits [14:10]

The destination register value from the issued instruction, containing the destination address.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

srcreg, bits [9:5]

The source register value from the issued instruction, containing either the source address or the source data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

sizereg, bits [4:0]

The size register value from the issued instruction, containing the number of bytes to be transferred or set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from HVC or SVC instruction execution

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										imm16														

Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
 - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
 - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, [HFGITR_EL2](#).{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				Op0		Op2		Op1		CRn			Rt			CRm			Direction					

Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write access, including MSR instructions.
0b1	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL1.
- [SCTLR_EL1](#).UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UCT, for accesses to [CTR_EL0](#) using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [CPACR_EL1](#).TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register, [ACTLR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.

- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TTRF, for accesses to the trace filter control register, [TRFCR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- [HCR_EL2](#).AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- [HCR_EL2](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.
- [SCR_EL3](#).APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TCPAC, for accesses to [CPTR_EL2](#) and [CPACR_EL1](#) using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TTRF, for accesses to the trace filter control registers, [TRFCR_EL1](#) and [TRFCR_EL2](#), using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
 - [HCR_EL2](#).{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
 - [HCR2](#).{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
 - [SCR_EL3](#).FGTE, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
 - [HFGTR_EL2](#) for reads and [HFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
 - [HFGITR_EL2](#) for execution of system instructions, MSR or MRS access trapped to EL2
 - [HDFGTR_EL2](#) for reads and [HDFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
 - [HAFGTR_EL2](#) for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
 - [SCR_EL3](#).TRNDR for reads of [RNDR](#) and [RNDRRS](#) using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:

- [CPTR_EL3](#).ESM, for MSR or MRS accesses to [SMPRI_EL1](#) at EL1, EL2, and EL3, trapped to EL3.
- [CPTR_EL3](#).ESM, for MSR or MRS accesses to [SMPRMAP_EL2](#) at EL2 and EL3, trapped to EL3.
- [SCTLR_EL1](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL1 or EL2.
- [SCTLR_EL2](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL2.
- [SCR_EL3](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, EL1, and EL2, trapped to EL3.
- If FEAT_NMI is implemented, [HCRX_EL2](#).TALLINT, for MSR writes of [ALLINT](#) at EL1, trapped to EL2.

ISS encoding for an exception from MSRR, an MRRS, Instruction or 128-bit System instruction execution in AArch64 state Abort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0		Op0SET		Op2FnV		Op1EA		CRnRES0		RtS1PTW		RES0		CRmIFSC		Direction								

Bits [24:22:13]

Reserved, RES0.

Op0, bits [21:20]

SET, bits [12:11]

When FEAT_RAS is implemented:

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The Op0 value from the issued instruction.

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

Otherwise:

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:6]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

Note

This value represents register pair of X[Rt:0], X[Rt:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk.

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or	When FEAT_RAS is

0b011111	hardware update of translation table, level 2. Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	not implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RAS is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address-size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRmFnV, bits[4:110]

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The FAR CRmnot valueValid, fromfor thea issuedsynchronous instruction.External abort other than a synchronous External abort on a translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DirectionEA, bit [09]

Indicates the direction of the trapped instruction.

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

Direction	Meaning
0b0	Write access, MSRR instructions.
0b1	Read access, MRRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from due ante InstructionSME Abortfunctionality

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												SET	SMTC	FnV	EA	RES0	S1PTW	RES0	IFSC					

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#).

WhenThe accesses covered by this trap include: FEAT_S1POE or FEAT_S2POE is implemented, if a memory access generates a Instruction Abort due to a Permission fault, the ISS2 encoding for an exception from an Instruction Abort includes further information about the exception.

Bits [24:133]

Reserved, RES0.

SETSMTC, bits [122:110]**When FEAT_RAS is implemented:**

SynchronousSME ErrorTrap Type.Code. WhenIdentifies IFSCthe isreason for instruction trapping. 0b010000, describes the PE error state after taking the Instruction Abort exception.

0b011	SME instruction trapped because PSTATE.ZA is 0.
SETSMTC	Meaning
0b000b000	Recoverable state (UER). Access to SME functionality trapped as a result of CPACR_EL1 .SMEN, CPTR_EL2 .SMEN, CPTR_EL2 .TSM, or CPTR_EL3 .ESM, that is not reported using EC 0b000000.
0b100b001	UncontainableAdvanced (UC).SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b110b010	RestartableSME stateinstruction (UEO).trapped because PSTATE.SM is 0.

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR** and **SMCR_EL1** System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN** and **CPTR_EL2.TSM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2** at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3** from all Exception levels and any Security state, trapped to EL3.

Otherwise:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or	When FEAT_RAS is not implemented

0b011110	hardware update of translation table, level 1. Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due from a SME Granule functionality Protection Check

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RES0																						SMTC			
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RES0				S2PTWInD				GPCSC				VNCR				RES0				CMS1PTWInR				xFSC	

Bits [24:22]

The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#).

Bits [24:3]

Reserved, RES0.

SMTC S2PTW, bits [2:0]

SMTC indicates whether Code the Identifies Granule the Protection reason Check exception was on an access made for instruction a trapping stage 2 translation table walk.

SMTC S2PTW	Meaning	Applies when
0b0000b0	Access Fault to not SME on functionality a trapped stage as 2 a translation result table of walk. CPACR_EL1.SMEN , CPTR_EL2.SMEN , CPTR_EL2.TSM , or CPTR_EL3.ESM , that is not reported using EC 0b000000.	
0b0010b1	Advanced Fault SIMD, on SVE, a or stage SVE2 instruction translation trapped table because PSTATE.SM is 1. walk.	
0b010	SME instruction trapped because PSTATE.SM is 0.	
0b011	SME instruction trapped because PSTATE.ZA is 0.	
0b100	Access to the SME2 ZT0 register trapped as a result of SMCR_EL1.EZT0 , SMCR_EL2.EZT0 , or SMCR_EL3.EZT0 .	When FEAT_SME2 is implemented

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

All The other reset values behavior are of reserved. this field is:

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

InD	Meaning
0b0	Data access.
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

GPCSC	Meaning
0b000000	GPT address size fault at level 0.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction.

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk.

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

xFSC	Meaning	Applies when
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR** and **SMCR_EL1** System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN** and **CPTR_EL2.TSM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2** at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3** from all Exception levels and any Security state, trapped to EL3.
- If FEAT_SME2 is implemented:
 - SMCR_EL1.EZT0**, for accesses to ZT0 at EL1 and EL0, trapped to EL1 or EL2.
 - SMCR_EL2.EZT0**, for accesses to ZT0 at EL2, EL1, and EL0, trapped to EL2.
 - SMCR_EL3.EZT0**, for accesses to ZT0 at any Exception level, trapped to EL3.

ISS encoding for an exception from a GranuleData Protection CheckAbort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	S2PTW	Ind					GPCSC			VNCR		RES0		CM	S1PTW	WnR								
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE				SRT			Bit[15]	AR	VNCR	Bits[12:11]	FnV	EA	CM	S1PTW	WnR							DFSC

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
 - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
 - Is not performing register writeback.
 - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]**When ISV == 1:**

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:**Bits [24:22]**

Reserved, RES0.

Otherwise:

Reserved, RES0.

Bit[15]**When ISV == 1:****SF, bit [0] of bit [15]**

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
0b0	Instruction loads/stores a 32-bit general-purpose register.
0b1	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0 and FEAT_SME is implemented:

FnP, bit [0] of bit [15]

FAR not Precise.

FnP	Meaning
0b0	The FAR holds the faulting virtual address that generated the Data Abort.
0b1	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction in Streaming SVE mode, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AR, bit [14]

When ISV == 1:

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S2PTW, bit [21]**SSE, bit [21]****When ISV == 1:**

Indicates Syndrome whether Sign the Extend. Granule For Protection a Check byte, exception halfword, was or on word an load access operation, made indicates for whether a the stage data 2 item translation must table be walk sign extended.

S2PTWSSE	Meaning
0b0	Fault Sign-extension not on a stage 2 translation table walk required.
0b1	Fault Data on item a must stage be 2 translation table walk sign extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]**Otherwise:**

Indicates whether the Granule Protection Check exception was on an instruction or data access.

Reserved, RES0.

InD	Meaning
0b0	Data access.
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]**SRT, bits [20:16]****When ISV == 1:**

GranuleSyndrome ProtectionRegister CheckTransfer. StatusThe Code.register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

GPCSC	Meaning
0b000000	GPT address size fault at level 0.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LST, bits [1:0] of bits [12:11]

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	
0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RAS is implemented and DFSC == 0b010000:

SET, bits [1:0] of bits [12:11]

Synchronous Error Type. Used when a Synchronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Bits [12:9]

Bits[12:11]

When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

IndicatesFor a stage 2 fault, indicates whether the Granulefault Protectionwas Checka exceptionstage was2 fault on an access for stage 2 translationmade for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSCDFSC, bits [5:0]

Instruction or Data Fault Status Code.

0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented

0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
xFSCDFSC	Meaning	Applies when
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The **lookup** level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE				SRT			Bit[15]	AR	VNCR	Bits[12:11]	FnV	EACM	S1PTW	WnR						DFSC		

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_S1POE is implemented, if a memory access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

When FEAT_S1PIE is implemented, if a memory write access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
 - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
 - Is not performing register writeback.
 - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FnV is 0 and FAR_ELx is valid.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSE, bit [21]

When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

SSE	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SRT, bits [20:16]

When ISV == 1:

Syndrone Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[15]

When ISV == 1:

SF, bit [0] of bit [15]

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
0b0	Instruction loads/stores a 32-bit general-purpose register.
0b1	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0:

FnP, bit [0] of bit [15]

FAR not Precise.

FnP	Meaning	Applies when
0b0	The FAR holds the faulting virtual address that generated the Data Abort.	
0b1	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).	When FEAT_SME is implemented or FEAT_SVE is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AR, bit [14]

When ISV == 1:

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits[12:11]**When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:****LST, bits [1:0] of bits [12:11]**

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	
0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RAS is implemented and DFSC == 0b010000:**SET, bits [1:0] of bits [12:11]**

Synchronous Error Type. Used when a Synchronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction.

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented

0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	TFV							RES0						VECITR	IDF	RES0	IXF	UFF	OFF	DZF	IOF			

Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

TFV	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

Note

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IDF	Meaning
0b0	Input denormal floating-point exception has not occurred.
0b1	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IXF	Meaning
0b0	Inexact floating-point exception has not occurred.
0b1	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

UFF	Meaning
0b0	Underflow floating-point exception has not occurred.
0b1	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

OFF	Meaning
0b0	Overflow floating-point exception has not occurred.
0b1	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

DZF	Meaning
0b0	Divide by Zero floating-point exception has not occurred.
0b1	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IOF	Meaning
0b0	Invalid Operation floating-point exception has not occurred.
0b1	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the [FPCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the [FPSCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDS	RES0										IESB	AET		EA	RES0		DFSC							

IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

IDS	Meaning
0b0	Bits [23:0] of the ISS field holds the fields described in this encoding.
Note If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.	
0b1	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

Note

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]**When FEAT_IESB is implemented and DFSC == 0b010001:**

Implicit error synchronization event.

IESB	Meaning
0b0	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
0b1	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AET, bits [12:10]**When FEAT_RAS is implemented and DFSC == 0b010001:**

Asynchronous Error Type.

Describes the PE error state after taking the SError interrupt exception.

AET	Meaning
0b000	Uncontainable (UC).
0b001	Unrecoverable state (UEU).
0b010	Restartable state (UEO).
0b011	Recoverable state (UER).
0b110	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Note

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA, bit [9]**When FEAT_RAS is implemented and DFSC == 0b010001:**

External abort type. Provides an IMPLEMENTATION DEFINED classification of External aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]**When FEAT_RAS is implemented:**

Data Fault Status Code.

DFSC	Meaning
0b000000	Uncategorized error.
0b010001	Asynchronous SError interrupt.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																			IFSC					

Bits [24:6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions'.
- For exceptions from AArch32, see 'Breakpoint exceptions' and 'Vector Catch exceptions'.

ISS encoding for an exception from a Software Step exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ISV		RES0																EX		IFSC					

ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

ISV	Meaning
0b0	EX bit is RES0.
0b1	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:7]

Reserved, RES0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

EX	Meaning
0b0	An instruction other than a Load-Exclusive instruction was stepped.
0b1	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Software Step exceptions'.

ISS encoding for an exception from a Watchpoint exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	WPT					WPTV		WPF	FnP	RES0	VNCR	RES0	FnV	RES0	CM	RES0	WnR	DFSC						

Bit [24]

Reserved, RES0.

WPT, bits [23:18]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint **number**. ~~number, 0 to 15 inclusive.~~

All other values are reserved.

Otherwise:

Reserved, RES0.

WPTV, bit [17]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint number Valid.

WPTV	Meaning	Applies when
0b0	The WPT field is invalid, and holds an UNKNOWN value.	When FEAT_SME is implemented and FEAT_Debugv8p9 is not implemented
0b1	The WPT field is valid, and holds the number of a watchpoint that triggered a Watchpoint exception.	

When a Watchpoint exception is triggered by a watchpoint match:

- If the PE sets any of FnV, FnP, or WPF to 1, then the PE sets WPTV to 1.
- If the PE sets all of FnV, FnP, and WPF to 0, then the PE sets WPTV to an IMPLEMENTATION DEFINED value, 0 or 1.

Otherwise:

Reserved, RES0.

WPF, bit [16]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint might be false-positive.

WPF	Meaning	Applies when
0b0	The watchpoint matched the original address of the access or set of contiguous accesses.	
0b1	The watchpoint matched an access or set of contiguous accesses where the lowest accessed address was rounded down to the nearest multiple of 16 bytes and the highest accessed address was rounded up to the nearest multiple of 16 bytes minus 1, but the watchpoint might not have matched the original address of the access or set of contiguous accesses.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

FnP, bit [15]**When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:**

FAR not Precise.

This field only has meaning if the FAR is valid; that is, when the FnV field is 0. If the FnV field is 1, the FnP field is 0.

FnP	Meaning	Applies when
0b0	If the FnV field is 0, the FAR holds the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	
0b1	The FAR holds any address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

Bit [14]

Reserved, RES0.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the watchpoint came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.
0b1	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

FnV, bit [10]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

FAR not Valid.

FnV	Meaning	Applies when
0b0	The FAR is valid, and its value is as described by the FnP field.	
0b1	The FAR is invalid, and holds an UNKNOWN value.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

Bit [9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance instruction:

CM	Meaning
0b0	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Watchpoint exception was generated by the execution of a cache maintenance instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Watchpoint exception caused by an instruction reading from a memory location.
0b1	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										Comment														

Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from a TSTART instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0															Rd			RES0						

Bits [24:10]

Reserved, RES0.

Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

ISS encoding for an exception from Branch Target Identification instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																						BTTYPE		

Bits [24:2]

Reserved, RES0.

BTTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																						Exception as a result of an Instruction key or a Data key		Exception as a result of an A key or a B key

Bits [24:2]

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

Meaning	
0b0	Instruction Key.
0b1	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

Meaning	
0b0	A key.
0b1	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.
- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETA, ERETB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing ESR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic ESR_EL1 or ESR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x138];
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL2;
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL1;

```

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x138] = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t, 64];

```

MRS <Xt>, ESR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x138];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL1;
    else
        UNDEFINED;

```

MSR ESR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x138] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ESR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL2;

```

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        ESR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ESR_EL2 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL2.

Configuration

AArch64 System register ESR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HSR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

ESR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								ISS2																							
EC						IL		ISS																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ESR_EL2 is made UNKNOWN as a result of an exception return from EL2.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL2, the value of ESR_EL2 is UNKNOWN. The value written to ESR_EL2 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:56~~37~~]

Reserved, RES0.

Otherwise:

~~Reserved, RES0.~~

ISS2, bits [55:32]

ISS2, bits [36:32]

When FEAT LS64 is implemented:

ISS2 encoding for an exception, the bit assignments are:

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

ISS2 encoding for an exception from a Data Abort (EC == 0b100100 or EC == 0b100101)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														RES0	TagAccess	RES0	AssuredOnly	Overlay	DirtyBit			Xs	

Bits [23:10]

Reserved, RES0.

TagAccess, bit [9]

When FEAT_MTE_PERM is implemented:

NoTagAccess fault.

If a memory access generates a Data Abort for a Permission fault, this field indicates whether the fault is due to the NoTagAccess memory attribute.

TagAccess	Meaning
0b0	Permission fault is not due to the NoTagAccess memory attribute.
0b1	Permission fault is due to the NoTagAccess memory attribute.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [8]

Reserved, RES0.

AssuredOnly, bit [7]

When FEAT_THE is implemented:

AssuredOnly flag.

If a memory access generates a Stage 2 Data Abort, this field holds information about the fault.

AssuredOnly	Meaning
0b0	The Data Abort is not due to AssuredOnly.
0b1	The Data Abort is due to AssuredOnly.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [6]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag.

If a memory access generates a Data Abort for a Permission fault, this field holds information about the fault.

Overlay	Meaning
0b0	The Data Abort is due to Base Permissions.
0b1	The Data Abort is due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DirtyBit, bit [5]**When FEAT_S1PIE is implemented or FEAT_S2PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to dirty state.
0b1	Permission Fault is due to dirty state.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Xs, bits [4:0]**When FEAT_LS64 is implemented:**

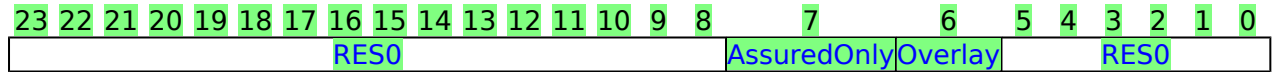
When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

Otherwise:

Reserved, RES0.

ISS2 encoding for an exception from an Instruction Abort (EC == 0b100000 or EC == 0b100001)**Bits [23:8]**

Reserved, RES0.

AssuredOnly, bit [7]**When FEAT_THE is implemented:**

AssuredOnly flag.

If a memory access generates a Instruction Abort for a Permission fault, then this field holds information about the fault.

AssuredOnly	Meaning
0b0	Instruction Abort is not due to AssuredOnly.
0b1	Instruction Abort is due to stage 2 AssuredOnly attribute.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [6]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag.

If a memory access generates a Instruction Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Instruction Abort is not due to Overlay Permissions.
0b1	Instruction Abort due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:0]

Reserved, RES0.

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

EC	Meaning	ISS	Applies when
0b000000	Unknown reason.	ISS encoding for exceptions with an unknown reason	
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS encoding for an exception from a WF* instruction	
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCRR or MRRC access	When AArch32 is supported
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: <ul style="list-style-type: none"> An STC to write data to memory from DBGDTRRXint. An LDC to read data from memory to DBGDTRTXint. 	ISS encoding for an exception from an LDC or STC instruction	When AArch32 is supported
0b000111	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN , CPTR_EL2.FPEN , CPTR_EL2.TFP , or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.	ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality resulting from the FPEN and TFP traps	
0b001000	Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b001001	Trapped use of a Pointer	ISS encoding for an exception	When FEAT_PAuth

	authentication instruction because HCR_EL2.API == 0 SCR_EL3.API == 0 .	from a Pointer Authentication instruction when HCR_EL2.API == 0 SCR_EL3.API == 0	is implemented
0b001010	An exception from an LD64B or ST64B* instruction.	ISS encoding for an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS encoding for an exception from an MCRR or MRRC access	When AArch32 is supported
0b001101	Branch Target Exception.	ISS encoding for an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
0b001110	Illegal Execution state.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b010001	SVC instruction execution in AArch32 state. This is reported in ESR_EL2 only when the exception is generated because the value of HCR_EL2.TGE is 1.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch32 is supported
0b010010	HVC instruction execution in AArch32 state, when HVC is not disabled.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch32 is supported
0b010011	SMC instruction execution in AArch32 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because the value of HCR_EL2.TSC is 1.	ISS encoding for an exception from SMC instruction execution in AArch32 state	When AArch32 is supported
0b010101	SVC instruction execution in AArch64 state.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch64 is supported
0b010110	HVC instruction execution in AArch64 state, when HVC is not disabled.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch64 is supported
0b010111	SMC instruction execution in AArch64 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because	ISS encoding for an exception from SMC instruction execution in AArch64 state	When AArch64 is supported

0b011000	<p>the value of HCR_EL2.TSC is 1. Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.</p>	ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
0b011001	<p>Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.</p>	ISS encoding for an exception from an access to SVE functionality resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented
0b011010	<p>Trapped ERET, ERETAA, or ERETAB instruction execution.</p>	ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction	When FEAT_PAuth is implemented and FEAT_NV is implemented
0b011011	<p>Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.</p>	ISS encoding for an exception from a TSTART instruction	When FEAT_TME is implemented
0b011100	<p>Exception from a Pointer Authentication instruction authentication failure</p>	ISS encoding for an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
0b011101	<p>Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM,</p>	ISS encoding for an exception due to SME functionality	When FEAT_SME is implemented

	CPTR_EL3 .ESM, or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.		
0b011110	Exception from a Granule Protection Check	ISS encoding for an exception from a Granule Protection Check	When FEAT_RME is implemented
0b100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from an Instruction Abort	
0b100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from an Instruction Abort	
0b100010	PC alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100100	Data Abort exception from a lower Exception level, excluding Data Abort exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. These Data Abort exceptions might be generated from Exception levels in any Execution state. Used for MMU faults generated by data accesses, alignment faults other than those	ISS encoding for an exception from a Data Abort	

	caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.		
0b100101	Data Abort exception without a change in Exception level, or Data Abort exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from a Data Abort	
0b100110	SP alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100111	Memory Operation Exception.	ISS encoding for an exception from the Memory Copy and Memory Set instructions	When FEAT_MOPS is implemented
0b101000	Trapped floating-point exception taken from AArch32 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS encoding for an exception from a trapped floating-point exception	When AArch32 is supported

0b101100	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.	ISS encoding for an exception from a trapped floating-point exception	When AArch64 is supported
0b101111	SError interrupt.	ISS encoding for an SError interrupt	
0b110000	Breakpoint exception from a lower Exception level.	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception	
0b110001	Breakpoint exception taken without a change in Exception level.	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception	
0b110010	Software Step exception from a lower Exception level.	ISS encoding for an exception from a Software Step exception	
0b110011	Software Step exception taken without a change in Exception level.	ISS encoding for an exception from a Software Step exception	
0b110100	Watchpoint from a lower Exception level, excluding Watchpoint Exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. These Watchpoint Exceptions might be generated from Exception levels using any Execution state.	ISS encoding for an exception from a Watchpoint exception	
0b110101	Watchpoint exceptions without a change in Exception level, or Watchpoint exceptions taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested	ISS encoding for an exception from a Watchpoint exception	

0b111000	virtualization support. BKPT instruction execution in AArch32 state.	ISS encoding for an exception from execution of a Breakpoint instruction	When AArch32 is supported
0b111010	Vector Catch exception from AArch32 state. The only case where a Vector Catch exception is taken to an Exception level that is using AArch64 is when the exception is routed to EL2 and EL2 is using AArch64.	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception	When AArch32 is supported
0b111100	BRK instruction execution in AArch64 state.	ISS encoding for an exception from execution of a Breakpoint instruction	When AArch64 is supported
0b111101	PMU exception	ISS encoding for a PMU exception	When FEAT_EBEP is implemented

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

IL	Meaning
0b0	16-bit instruction trapped.
0b1	32-bit instruction trapped. This value is also used when the exception is one of the following: <ul style="list-style-type: none"> • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: <ul style="list-style-type: none"> ◦ 0b0: 16-bit T32 BKPT instruction. ◦ 0b1: 32-bit A32 BKPT instruction or A64 BRK instruction. • An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS, bits [24:0]

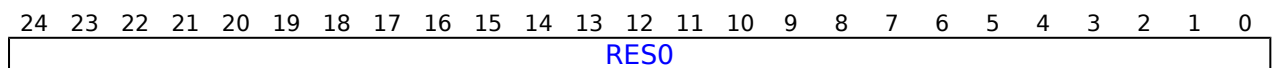
Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
 - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
 - The value 0b11111.

ISS encoding for exceptions with an unknown reason**Bits [24:0]**

Reserved, RES0.

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
 - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
 - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
 - Instruction encodings that are unallocated.
 - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the [SCTLR_EL1](#).{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
 - An HVC instruction when disabled by [HCR_EL2](#).HCD or [SCR_EL3](#).HCE.
 - An SMC instruction when disabled by [SCR_EL3](#).SMD.
 - An HLT instruction when disabled by [EDSCR](#).HDE.
- Attempted execution of an MSR or MRS instruction to access [SP_EL0](#) when the value of [SPSel](#).SP is 0.
- Attempted execution of an MSR or MRS instruction using a [_EL12](#) register name when [HCR_EL2](#).E2H == 0.
- Attempted execution, in Debug state, of:
 - A DCPS1 instruction when the value of [HCR_EL2](#).TGE is 1 and EL2 is disabled or not implemented in the current Security state.
 - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.

- A DCPS3 instruction when the value of [EDSCR.SDD](#) is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13_mon.
- In Debug state when the value of [EDSCR.SDD](#) is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to SPSR_mon, SP_mon, or LR_mon.
- An exception that is taken to EL2 because the value of [HCR_EL2.TGE](#) is 1 that, if the value of [HCR_EL2.TGE](#) was 0 would have been reported with an ESR_ELx.EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF* instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0										RN			RES0		RV	TI			

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]**When FEAT_WFxT is implemented:**

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [4:3]

Reserved, RES0.

RV, bit [2]**When FEAT_WFxT is implemented:**

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

RV	Meaning
0b0	Register field invalid.
0b1	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

TI	Meaning	Applies when
0b00	WFI trapped.	
0b01	WFE trapped.	
0b10	WFIT trapped.	When FEAT_WFxT is implemented
0b11	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating this exception:

- [SCTLR_EL1](#).{nTWE, nTWI}.
- [HCR_EL2](#).{TWE, TWI}.
- [SCR_EL3](#).{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc2		Opc1		CRn				Rt				CRm				Direction			

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCR instruction.
0b1	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#) or [CPACR](#) using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CNTHCTL_EL2](#).EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL3](#).TCPAC, for accesses to [CPACR](#) from EL1 and EL2, and accesses to [HCPTR](#) from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- [CPACR_EL1](#).TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [HCR_EL2](#).TID0, for accesses to the [JIDR](#) register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.

- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- [MDCR_EL3](#).TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- [HCR_EL2](#).TID0, for accesses to the [FPSID](#) register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- [HCR_EL2](#).TID3, for accesses to registers in ID group 3 including [MVFR0](#), [MVFR1](#) and [MVFR2](#), VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B* instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISS																								

ISS, bits [24:0]

ISS	Meaning	Applies when
0b000000000000000000000000000000	ST64BV instruction trapped.	When FEAT_LS64_V is implemented
0b000000000000000000000000000001	ST64BV0 instruction trapped.	When FEAT_LS64_ACCDATA is implemented
0b000000000000000000000000000010	LD64B or ST64B instruction trapped.	When FEAT_LS64 is implemented

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc1				RES0	Rt2				Rt				CRm				Direction		

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.

- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b1111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b1111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCRR instruction.
0b1	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.

- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, [HDFGRTR_EL2](#).PMCCNTR_EL0 for MRRC access and [HDFGWTR_EL2](#).PMCCNTR_EL0 for MCRR access to [PMCCNTR](#) at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- [MDSCR_EL1](#).TDCC, for accesses to the Debug ROM registers [DBGDSAR](#) and [DBGDRAR](#) at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- [CPACR_EL1](#).TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

Note

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				imm8								RES0		Rn				Offset		AM		Direction	

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b1111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b1111.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Offset	Meaning
0b0	Subtract offset.
0b1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
0b000	Immediate unindexed.
0b001	Immediate post-indexed.
0b010	Immediate offset.
0b011	Immediate pre-indexed.
0b100	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to memory. STC instruction.
0b1	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- [MDSCR_EL1](#).TDCC, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPE and TFP traps

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0																			

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

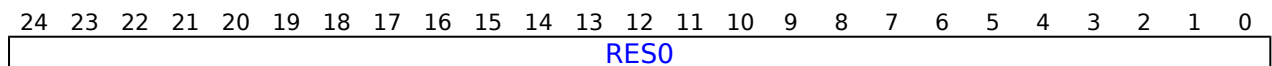
Bits [19:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- [CPACR_EL1.FPEN](#), for accesses to SIMD and floating-point registers trapped to EL1.
- [CPTR_EL2.FPEN](#) and [CPTR_EL2.TFP](#), for accesses to SIMD and floating-point registers trapped to EL2.
- [CPTR_EL3.TFP](#), for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ



The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

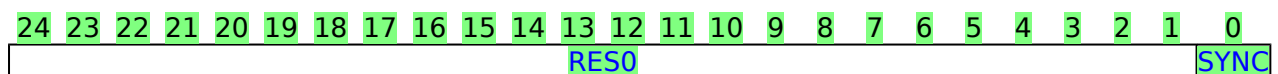
Bits [24:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- [CPACR_EL1.ZEN](#), for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- [CPTR_EL2.ZEN](#) and [CPTR_EL2.TZ](#), for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- [CPTR_EL3.EZ](#), for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for a PMU exception

**Bits [24:1]**

Reserved, RES0.

SYNC, bit [0]

Indicates whether the exception was taken synchronously or asynchronously.

SYNC	Meaning	Applies when
0b0	The exception was taken asynchronously because an overflow status flag was set.	
0b1	The exception was taken synchronously because PSTATE.PPEND was set.	When FEAT_SEBEP is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								

Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from the Memory Copy and Memory Set instructions

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																									
MemInst					isSETG					Options					FromEpilogue					WrongOption					OptionA					RES0					destreg					srcreg					sizereg				

MemInst, bit [24]

Indicates the memory instruction class causing the exception.

MemInst	Meaning
0b0	CPYFE*, CPYFM*, CPYE*, and CPYM* instructions.
0b1	SETE*, SETM*, SETGE*, and SETGM* instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

isSETG, bit [23]

Indicates whether the instruction belongs to SETGM* or SETGE* class of instruction.

isSETG	Meaning
0b0	Not a SETGM* or SETGE* instruction.
0b1	SETGM* or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Options, bits [22:19]

Options : the Options field of the instruction.

For Memory Copy instructions, bits[22:19] forms the Options field, which holds the bits[15:12] of the instruction.

For Memory Set instructions:

- Bits[22:21] are RES0.
- Bits[20:19] form the Options field, which holds the bits[13:12] of the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FromEpilogue, bit [18]

Indicates whether the instruction belongs to the epilogue class of Memory Copy or Memory Set instructions.

FromEpilogue	Meaning
0b0	Not an epilogue instruction.
0b1	CPYE*, CPYFE*, SETE*, or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WrongOption, bit [17]

Algorithm option.

WrongOption	Meaning
0b0	WrongOption is false.
0b1	WrongOption is true.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OptionA, bit [16]

Algorithm type indicated by the PSTATE.C bit.

OptionA	Meaning
0b0	OptionB indicated by PSTATE.C is 0.
0b1	OptionA indicated by PSTATE.C is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

destreg, bits [14:10]

The destination register value from the issued instruction, containing the destination address.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

srcreg, bits [9:5]

The source register value from the issued instruction, containing either the source address or the source data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

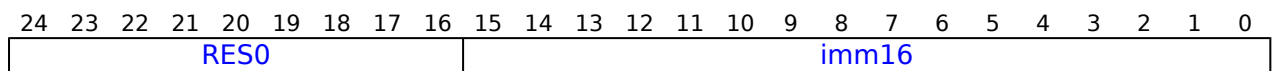
sizereg, bits [4:0]

The size register value from the issued instruction, containing the number of bytes to be transferred or set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from HVC or SVC instruction execution

**Bits [24:16]**

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
 - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
 - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, [HFGITR_EL2](#).{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from SMC instruction execution in AArch32 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				CCKNOWNPASS				RES0															

For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from EL1 because [HCR_EL2.TSC](#) is 1, the ISS encoding is as shown in the diagram.

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

CCKNOWNPASS	Meaning
0b0	The instruction was unconditional, or was conditional and passed its condition code check.
0b1	The instruction was conditional, and might have failed its condition code check.

Note

In an implementation in which an SMC instruction that fails its code check is not trapped, this field can always return the value 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [18:0]

Reserved, RES0.

[HCR_EL2](#).TSC describes the configuration settings for trapping SMC instructions to EL2.

ISS encoding for an exception from SMC instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										imm16														

Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the issued SMC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The value of ISS[24:0] described here is used both:

- When an SMC instruction is trapped from EL1 modes.
- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

[HCR_EL2](#).TSC describes the configuration settings for trapping SMC from EL1 modes.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RES0		Op0		Op2		Op1		CRn		Rt		CRm		Direction											

Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write access, including MSR instructions.
0b1	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL1.
- [SCTLR_EL1](#).UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UCT, for accesses to [CTR_EL0](#) using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [CPACR_EL1](#).TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TLTB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register, [ACTLR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TTRF, for accesses to the trace filter control register, [TRFCR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- [HCR_EL2](#).AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.

- [HCR_EL2](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.
- [SCR_EL3](#).APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TCPAC, for accesses to [CPTR_EL2](#) and [CPACR_EL1](#) using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TTRF, for accesses to the trace filter control registers, [TRFCR_EL1](#) and [TRFCR_EL2](#), using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
 - [HCR_EL2](#).{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
 - [HCR2](#).{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
 - [SCR_EL3](#).FGTE, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
 - [HFGTR_EL2](#) for reads and [HFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
 - [HFGITR_EL2](#) for execution of system instructions, MSR or MRS access trapped to EL2
 - [HDFGTR_EL2](#) for reads and [HDFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
 - [HAFGTR_EL2](#) for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
 - [SCR_EL3](#).TRNDR for reads of [RNDR](#) and [RNDRRS](#) using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
 - [CPTR_EL3](#).ESM, for MSR or MRS accesses to [SMPRI_EL1](#) at EL1, EL2, and EL3, trapped to EL3.
 - [CPTR_EL3](#).ESM, for MSR or MRS accesses to [SMPRMAP_EL2](#) at EL2 and EL3, trapped to EL3.
 - [SCTLR_EL1](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL1 or EL2.
 - [SCTLR_EL2](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL2.
 - [SCR_EL3](#).EnTP2, for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, EL1, and EL2, trapped to EL3.
- If FEAT_NMI is implemented, [HCRX_EL2](#).TALLINT, for MSR writes of [ALLINT](#) at EL1, trapped to EL2.

ISS encoding for an exception from MSRR, an MRRS, Instruction or 128-bit System instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0		Op0SET		Op2FnV		Op1EA		CRnRES0		RtS1PTW		RES0		CRmIFSC		Direction								

Bits [24:22:13]

Reserved, RES0.

Op0, bits [21:20]**SET, bits [12:11]****When FEAT_RAS is implemented:**

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The Op0 value from the issued instruction.

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]**Otherwise:**

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:6]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

Note

This value represents register pair of X[Rt:0], X[Rt:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or	When FEAT_RAS is

0b011111	hardware update of translation table, level 2. Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	not implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RAS is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address-size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRmFnV, bitsbit [4:110]

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The FAR CRmnot valueValid, fromfor thea issuedsynchronous instruction.External abort other than a synchronous External abort on a translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DirectionEA, bit [09]

Indicates the direction of the trapped instruction.

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

Direction	Meaning
0b0	Write access, MSRR instructions.
0b1	Read access, MRRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from due ante InstructionSME Abortfunctionality

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	TopLevelSMTC			RES0				SET	FnV	EARES0	S1PTW	RES0										IFSC		

WhenThe accesses covered by this trap include: FEAT_THE is implemented, if a memory access generates an Instruction Abort for AssuredOnly, the ISS2 encoding for an exception from an Instruction Abort includes further information about the exception.

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#).

When FEAT_S1POE or FEAT_S2POE is implemented, if a memory access generates an Instruction Abort due to a Permission fault, the ISS2 encoding for an exception from an Instruction Abort includes further information about the exception.

Bits [24:223]

Reserved, RES0.

TopLevelSMTC, bitbits [212:0]**When FEAT_THE is implemented:**

IndicatesSME ifTrap Code. Identifies the faultreason wasfor dueinstruction to TopLevel.trapping.

0b010	SME instruction trapped because PSTATE.SM is 0.
0b011	SME instruction trapped because PSTATE.ZA is 0.
TopLevelSMTC	Meaning
0b00b000	Fault is not due to VTCR_EL2.TL0 or VTCR_EL2.TL1 Access to SME functionality trapped as a result of CPACR_EL1.SMEN , CPTR_EL2.SMEN , CPTR_EL2.TSM , or CPTR_EL3.ESM , that is not reported using EC 0b000000.
0b10b001	FaultAdvanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is due to 1, VTCR_EL2.TL0 or VTCR_EL2.TL1

TheAll resetother behaviorvalues ofare this field is:reserved.

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- **CPACR_EL1.SMEN**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR** and **SMCR_EL1** System registers at EL1 and EL0, trapped to EL1 or EL2.
- **CPTR_EL2.SMEN** and **CPTR_EL2.TSM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2** at EL2, EL1, or EL0, trapped to EL2.
- **CPTR_EL3.ESM**, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access **SVCR**, **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3** from all Exception levels and any Security state, trapped to EL3.

Otherwise:

Reserved, RES0.

Bits [20:13]

Reserved, RES0.

SET, bits [12:11]**When FEAT_RAS is implemented and IFSC == 0b010000:**

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnV, bit [10]**When IFSC == 0b010000:**

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or	When FEAT_RAS is not implemented

0b011110	hardware update of translation table, level 1. Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

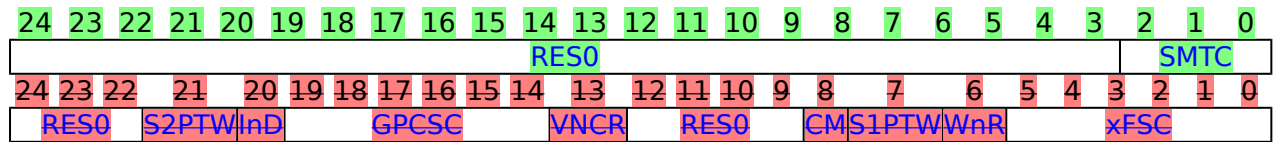
For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due from a SME Granule functionality Protection Check



Bits [24:22]

The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#).

Bits [24:3]

Reserved, RES0.

SMTC S2PTW, bits [2:0]

SMTC indicates whether Code the Identifies Granule the Protection reason Check exception was on an access made for instruction a trapping stage 2 translation table walk.

SMTC S2PTW	Meaning	Applies when
0b0000b0	Access Fault to not SME on functionality a trapped stage as 2 a translation result table of walk. CPACR_EL1.SMEN , CPTR_EL2.SMEN , CPTR_EL2.TSM , or CPTR_EL3.ESM , that is not reported using EC 0b000000.	
0b0010b1	Advanced Fault SIMD, on SVE, a or stage SVE2 instruction translation trapped table because PSTATE.SM is 1. walk.	
0b010	SME instruction trapped because PSTATE.SM is 0.	
0b011	SME instruction trapped because PSTATE.ZA is 0.	
0b100	Access to the SME2 ZT0 register trapped as a result of SMCR_EL1.EZT0 , SMCR_EL2.EZT0 , or SMCR_EL3.EZT0 .	When FEAT_SME2 is implemented

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

All The other reset values behavior are of reserved. this field is:

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

InD	Meaning
0b0	Data access.
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

GPCSC	Meaning
0b000000	GPT address size fault at level 0.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2, by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction.

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk.

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

xFSC	Meaning	Applies when
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- CPACR_EL1.SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVC](#)R and [SMCR_EL1](#) System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN and CPTL_EL2.TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVC](#)R, [SMCR_EL1](#), [SMCR_EL2](#) at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVC](#)R, [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#) from all Exception levels and any Security state, trapped to EL3.
- If FEAT_SME2 is implemented:
 - [SMCR_EL1](#).EZT0, for accesses to ZT0 at EL1 and EL0, trapped to EL1 or EL2.
 - [SMCR_EL2](#).EZT0, for accesses to ZT0 at EL2, EL1, and EL0, trapped to EL2.
 - [SMCR_EL3](#).EZT0, for accesses to ZT0 at any Exception level, trapped to EL3.

ISS encoding for an exception from a GranuleData Protection CheckAbort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	S2PTW	Ind					GPCSC			VNCR		RES0		CM	S1PTW	WnR								
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE				SRT			Bit[15]	AR	VNCR	Bits[12:11]	FnV	EA	CM	S1PTW	WnR							DFSC

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this ISS encoding includes ISS2, bits[36:32].

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
 - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
 - Is not performing register writeback.
 - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]**When ISV == 1:**

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:**Bits [24:22]**

Reserved, RES0.

Otherwise:

Reserved, RES0.

Bit[15]**When ISV == 1:****SF, bit [0] of bit [15]**

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
0b0	Instruction loads/stores a 32-bit general-purpose register.
0b1	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0 and FEAT_SME is implemented:

FnP, bit [0] of bit [15]

FAR not Precise.

FnP	Meaning
0b0	The FAR holds the faulting virtual address that generated the Data Abort.
0b1	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction in Streaming SVE mode, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AR, bit [14]

When ISV == 1:

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S2PTW, bit [21]**SSE, bit [21]****When ISV == 1:**

Indicates Syndrome whether Sign the Extend. Granule For Protection a Check byte, exception halfword, was or on word an load access operation, made indicates for whether a the staged data 2 item translation must table be walk sign extended.

S2PTWSSE	Meaning
0b0	Fault Sign-extension not on a stage 2 translation table walk required.
0b1	Fault Data on item a must stage be 2 translation table walk sign extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]**Otherwise:**

Indicates whether the Granule Protection Check exception was on an instruction or data access.

Reserved, RES0.

InD	Meaning
0b0	Data access.
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]**SRT, bits [20:16]****When ISV == 1:**

GranuleSyndrome ProtectionRegister CheckTransfer. StatusThe Code.register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

GPCSC	Meaning
0b000000	GPT address size fault at level 0.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LST, bits [1:0] of bits [12:11]

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	
0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RAS is implemented and DFSC == 0b010000:

SET, bits [1:0] of bits [12:11]

Synchronous Error Type. Used when a Synchronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Bits [12:9]

Bits[12:11]

When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

IndicatesFor a stage 2 fault, indicates whether the Granulefault Protectionwas Checka exceptionstage was2 fault on an access for stage 2 translationmade for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSCDFSC, bits [5:0]

Instruction or Data Fault Status Code.

0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented

0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
xFSCDFSC	Meaning	Applies when
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The **lookup** level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	Bit[21]		SRT		Bit[15]	AR	VNCR	Bits[12:11]	FnV	EA	CM	S1PTW	WnR							DFSC			

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_THE is implemented, if a memory access generates a Data Abort for AssuredOnly, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

When FEAT_S1POE or FEAT_S2POE is implemented, if a memory access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

When FEAT_S1PIE or FEAT_S2PIE is implemented, if a memory write access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
 - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
 - Is not performing register writeback.
 - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[21]

When ISV == 1:

SSE, bit [0] of bit [21]

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

SSE	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0 and FEAT_THE is implemented:

TopLevel, bit [0] of bit [21]

Indicates if the fault was due to TopLevel.

TopLevel	Meaning
0b0	Fault is not due to TopLevel.
0b1	Fault is due to TopLevel.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SRT, bits [20:16]

When ISV == 1:

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[15]**When ISV == 1:****SF, bit [0] of bit [15]**

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
0b0	Instruction loads/stores a 32-bit general-purpose register.
0b1	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0:**FnP, bit [0] of bit [15]**

FAR not Precise.

FnP	Meaning	Applies when
0b0	The FAR holds the faulting virtual address that generated the Data Abort.	
0b1	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).	When FEAT_SME is implemented or FEAT_SVE is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AR, bit [14]**When ISV == 1:**

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits[12:11]**When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:****LST, bits [1:0] of bits [12:11]**

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	
0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RAS is implemented and DFSC == 0b010000:**SET, bits [1:0] of bits [12:11]**

Synchronous Error Type. Used when a Synchronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented

0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	TFV							RES0						VECITR	IDF	RES0	IXF	UFF	OFF	DZF	IOF			

Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

TFV	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

Note

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IDF	Meaning
0b0	Input denormal floating-point exception has not occurred.
0b1	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IXF	Meaning
0b0	Inexact floating-point exception has not occurred.
0b1	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

UFF	Meaning
0b0	Underflow floating-point exception has not occurred.
0b1	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

OFF	Meaning
0b0	Overflow floating-point exception has not occurred.
0b1	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

DZF	Meaning
0b0	Divide by Zero floating-point exception has not occurred.
0b1	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IOF	Meaning
0b0	Invalid Operation floating-point exception has not occurred.
0b1	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the [FPCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the [FPSCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDS	RES0										IESB	AET		EA	RES0		DFSC							

IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

IDS	Meaning
0b0	Bits [23:0] of the ISS field holds the fields described in this encoding.
Note If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.	
0b1	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

Note

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]**When FEAT_IESB is implemented and DFSC == 0b010001:**

Implicit error synchronization event.

IESB	Meaning
0b0	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
0b1	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AET, bits [12:10]**When FEAT_RAS is implemented and DFSC == 0b010001:**

Asynchronous Error Type.

Describes the PE error state after taking the SError interrupt exception.

AET	Meaning
0b000	Uncontainable (UC).
0b001	Unrecoverable state (UEU).
0b010	Restartable state (UEO).
0b011	Recoverable state (UER).
0b110	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Note

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA, bit [9]**When FEAT_RAS is implemented and DFSC == 0b010001:**

External abort type. Provides an IMPLEMENTATION DEFINED classification of External aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]**When FEAT_RAS is implemented:**

Data Fault Status Code.

DFSC	Meaning
0b000000	Uncategorized error.
0b010001	Asynchronous SError interrupt.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
RES0																				IFSC							

Bits [24:6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions'.
- For exceptions from AArch32, see 'Breakpoint exceptions' and 'Vector Catch exceptions'.

ISS encoding for an exception from a Software Step exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	RES0														EX		IFSC							

ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

ISV	Meaning
0b0	EX bit is RES0.
0b1	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:7]

Reserved, RES0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

EX	Meaning
0b0	An instruction other than a Load-Exclusive instruction was stepped.
0b1	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Software Step exceptions'.

ISS encoding for an exception from a Watchpoint exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	WPT					WPTV		WPF	FnP	RES0	VNCR	RES0	FnV	RES0	CM	RES0	WnR	DFSC						

Bit [24]

Reserved, RES0.

WPT, bits [23:18]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint **number**. ~~number, 0 to 15 inclusive.~~

All other values are reserved.

Otherwise:

Reserved, RES0.

WPTV, bit [17]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint number Valid.

WPTV	Meaning	Applies when
0b0	The WPT field is invalid, and holds an UNKNOWN value.	When FEAT_SME is implemented and FEAT_Debugv8p9 is not implemented
0b1	The WPT field is valid, and holds the number of a watchpoint that triggered a Watchpoint exception.	

When a Watchpoint exception is triggered by a watchpoint match:

- If the PE sets any of FnV, FnP, or WPF to 1, then the PE sets WPTV to 1.
- If the PE sets all of FnV, FnP, and WPF to 0, then the PE sets WPTV to an IMPLEMENTATION DEFINED value, 0 or 1.

Otherwise:

Reserved, RES0.

WPF, bit [16]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

Watchpoint might be false-positive.

WPF	Meaning	Applies when
0b0	The watchpoint matched the original address of the access or set of contiguous accesses.	
0b1	The watchpoint matched an access or set of contiguous accesses where the lowest accessed address was rounded down to the nearest multiple of 16 bytes and the highest accessed address was rounded up to the nearest multiple of 16 bytes minus 1, but the watchpoint might not have matched the original address of the access or set of contiguous accesses.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

FnP, bit [15]**When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:**

FAR not Precise.

This field only has meaning if the FAR is valid; that is, when the FnV field is 0. If the FnV field is 1, the FnP field is 0.

FnP	Meaning	Applies when
0b0	If the FnV field is 0, the FAR holds the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	
0b1	The FAR holds any address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

Bit [14]

Reserved, RES0.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the watchpoint came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.
0b1	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

FnV, bit [10]

When FEAT_SME is implemented or FEAT_Debugv8p9 is implemented:

FAR not Valid.

FnV	Meaning	Applies when
0b0	The FAR is valid, and its value is as described by the FnP field.	
0b1	The FAR is invalid, and holds an UNKNOWN value.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

Otherwise:

Reserved, RES0.

Bit [9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance instruction:

CM	Meaning
0b0	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Watchpoint exception was generated by the execution of a cache maintenance instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Watchpoint exception caused by an instruction reading from a memory location.
0b1	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										Comment														

Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																							ERET	ERETA

This EC value applies when FEAT_FGT is implemented, or when [HCR_EL2.NV](#) is 1.

Bits [24:2]

Reserved, RES0.

ERET, bit [1]

Indicates whether an ERET or ERETAA* instruction was trapped to EL2.

ERET	Meaning
0b0	ERET instruction trapped to EL2.
0b1	ERETAA or ERETAB instruction trapped to EL2.

If this bit is 0, the ERETA field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ERETA, bit [0]

Indicates whether an ERETAA or ERETAB instruction was trapped to EL2.

ERETA	Meaning
0b0	ERETAA instruction trapped to EL2.
0b1	ERETAB instruction trapped to EL2.

When the ERET field is 0, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see [HCR_EL2.NV](#).

If FEAT_FGT is implemented, [HFGITR_EL2.ERET](#) controls fine-grained trap exceptions from ERET, ERETAA and ERETAB execution.

ISS encoding for an exception from a TSTART instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0															Rd					RES0				

Bits [24:10]

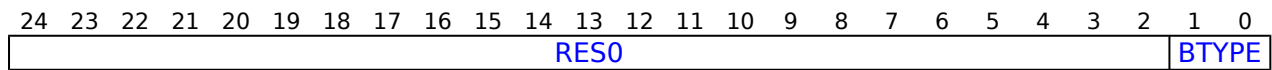
Reserved, RES0.

Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

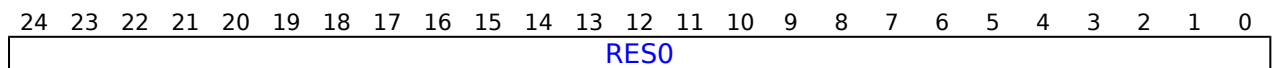
ISS encoding for an exception from Branch Target Identification instruction**Bits [24:2]**

Reserved, RES0.

BTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

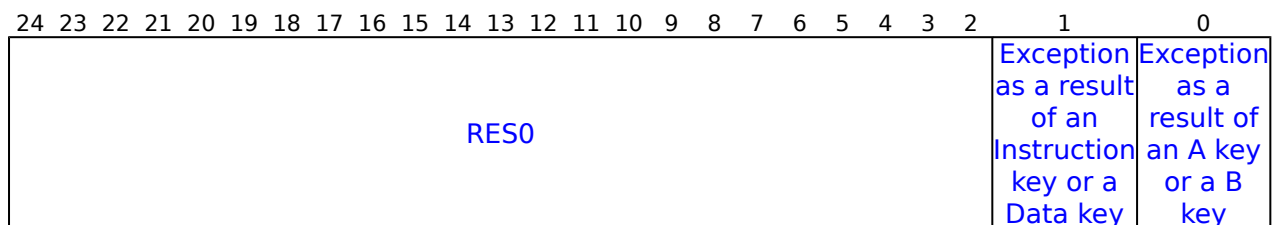
For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0**Bits [24:0]**

Reserved, RES0.

For more information about generating these exceptions, see:

- [HCR_EL2.API](#), for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2.
- [SCR_EL3.API](#), for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL3.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure**Bits [24:2]**

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

	Meaning
0b0	Instruction Key.
0b1	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

	Meaning
0b0	A key.
0b1	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.
- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing ESR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic ESR_EL2 or ESR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ESR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL2;

```

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        ESR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t, 64];

```

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x138];
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL2;
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL1;

```

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ESR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x138] = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ESR_EL3, Exception Syndrome Register (EL3)

The ESR_EL3 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to ESR_EL3 are UNDEFINED.

Attributes

ESR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0								ISS2																								
EC						IL	ISS																									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

ESR_EL3 is made UNKNOWN as a result of an exception return from EL3.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL3, the value of ESR_EL3 is UNKNOWN. The value written to ESR_EL3 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:5637]

Reserved, RES0.

Otherwise:

Reserved, RES0.

ISS2, bits [55:32]

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

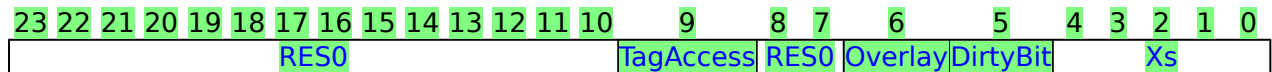
ISS2 encoding for an exception, the bit assignments are:

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

ISS2 encoding for an exception from a Data Abort (EC == 0b100100 or EC == 0b100101)



Bits [23:10]

Reserved, RES0.

TagAccess, bit [9]

When FEAT_MTE_PERM is implemented:

NoTagAccess fault.

If a memory access generates a Data Abort for a Permission fault, this field indicates whether the fault is due to the NoTagAccess memory attribute.

TagAccess	Meaning
0b0	Permission fault is not due to the NoTagAccess memory attribute.
0b1	Permission fault is due to the NoTagAccess memory attribute.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [8:7]

Reserved, RES0.

Overlay, bit [6]

When FEAT_S1POE is implemented:

Overlay flag.

If a memory access generates a Data Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Data Abort is not due to Overlay Permissions.
0b1	Data Abort due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DirtyBit, bit [5]**When FEAT_S1PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, then this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to dirty state.
0b1	Permission Fault is due to dirty state.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Xs, bits [4:0]**When FEAT_LS64 is implemented:**

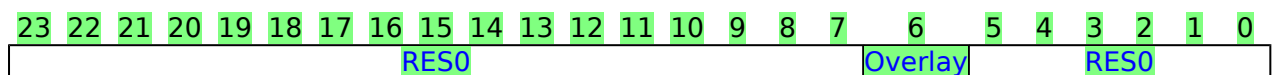
When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is RES0.

Otherwise:

Reserved, RES0.

ISS2 encoding for an exception from an Instruction Abort (EC == 0b100000 or EC == 0b100001)**Bits [23:7]**

Reserved, RES0.

Overlay, bit [6]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag.

If a memory access generates a Instruction Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Instruction Abort is not due to Overlay Permissions.
0b1	Instruction Abort due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:0]

Reserved, RES0.

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

EC	Meaning	ISS	Applies when
0b000000	Unknown reason.	ISS encoding for exceptions with an unknown reason	
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check do not cause an exception.	ISS encoding for an exception from a WF* instruction	
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for an exception from an MCRR or MRRC access	When AArch32 is supported
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS encoding for an exception from an MCR or MRC access	When AArch32 is supported
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: <ul style="list-style-type: none"> An STC to write data to memory from DBGDTRRXint. An LDC to read data from memory to DBGDTRTXint. 	ISS encoding for an exception from an LDC or STC instruction	When AArch32 is supported
0b000111	Access to SME, SVE, Advanced SIMD or floating-point functionality trapped by CPACR_EL1.FPEN , CPTR_EL2.FPEN , CPTR_EL2.TFP , or CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value of HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.	ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps	
0b001001	Trapped use of a Pointer authentication instruction because HCR_EL2.API == 0 SCR_EL3.API == 0.	ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 SCR_EL3.API == 0	When FEAT_PAuth is implemented

0b001010	An exception from an LD64B or ST64B* instruction.	ISS encoding for an exception from an LD64B or ST64B* instruction	When FEAT_LS64 is implemented
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS encoding for an exception from an MCRR or MRRC access	When AArch32 is supported
0b001101	Branch Target Exception.	ISS encoding for an exception from Branch Target Identification instruction	When FEAT_BTI is implemented
0b001110	Illegal Execution state.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b010011	SMC instruction execution in AArch32 state, when SMC is not disabled.	ISS encoding for an exception from SMC instruction execution in AArch32 state	When AArch32 is supported
0b010101	SVC instruction execution in AArch64 state.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch64 is supported
0b010110	HVC instruction execution in AArch64 state, when HVC is not disabled.	ISS encoding for an exception from HVC or SVC instruction execution	When AArch64 is supported
0b010111	SMC instruction execution in AArch64 state, when SMC is not disabled.	ISS encoding for an exception from SMC instruction execution in AArch64 state	When AArch64 is supported
0b011000	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111. This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state	When AArch64 is supported
0b011001	Access to SVE functionality trapped as a result of CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000.	ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ	When FEAT_SVE is implemented

0b011011	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at EL1 when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.	ISS encoding for an exception from a TSTART instruction	When FEAT_TME is implemented
0b011100	Exception from a Pointer Authentication instruction authentication failure	ISS encoding for an exception from a Pointer Authentication instruction authentication failure	When FEAT_FPAC is implemented
0b011101	Access to SME functionality trapped as a result of CPACR_EL1.SMEN , CPTR_EL2.SMEN , CPTR_EL2.TSM , CPTR_EL3.ESM , or an attempted execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b000000.	ISS encoding for an exception due to SME functionality	When FEAT_SME is implemented
0b011110	Exception from a Granule Protection Check	ISS encoding for an exception from a Granule Protection Check	When FEAT_RME is implemented
0b011111	IMPLEMENTATION DEFINED exception to EL3.	ISS encoding for an IMPLEMENTATION DEFINED exception to EL3	
0b100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from an Instruction Abort	
0b100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including	ISS encoding for an exception from an Instruction Abort	

0b100010	synchronous parity or ECC errors. Not used for debug-related exceptions. PC alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100100	Data Abort exception from a lower Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from a Data Abort	
0b100101	Data Abort exception taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.	ISS encoding for an exception from a Data Abort	
0b100110	SP alignment fault exception.	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault	
0b100111	Memory Operation Exception.	ISS encoding for an exception from the Memory Copy and Memory Set instructions	When FEAT_MOPS is implemented
0b101100	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping of floating-point exceptions, otherwise it is reserved. Whether a floating-point	ISS encoding for an exception from a trapped floating-point exception	When AArch64 is supported

	implementation supports trapping of floating-point exceptions is IMPLEMENTATION DEFINED.		
0b101111	SError interrupt.	ISS encoding for an SError interrupt	
0b111100	BRK instruction execution in AArch64 state. This is reported in ESR_EL3 only if a BRK instruction is executed in EL3. This is the only debug exception that can be taken to EL3 when EL3 is using AArch64.	ISS encoding for an exception from execution of a Breakpoint instruction	When AArch64 is supported
0b111101	PMU exception	ISS encoding for a PMU exception	When FEAT_EBEP is implemented

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

IL	Meaning
0b0	16-bit instruction trapped.
0b1	32-bit instruction trapped. This value is also used when the exception is one of the following: <ul style="list-style-type: none"> • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. • An exception reported using EC value 0b000000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

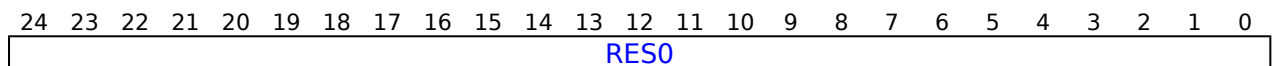
Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
 - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
 - The value 0b11111.

ISS encoding for exceptions with an unknown reason



Bits [24:0]

Reserved, RES0.

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
 - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
 - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
 - Instruction encodings that are unallocated.
 - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the [SCTLR_EL1](#).{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
 - An HVC instruction when disabled by [HCR_EL2](#).HCD or [SCR_EL3](#).HCE.
 - An SMC instruction when disabled by [SCR_EL3](#).SMD.
 - An HLT instruction when disabled by [EDSCR](#).HDE.
- Attempted execution of an MSR or MRS instruction to access [SP_EL0](#) when the value of [SPSel](#).SP is 0.
- Attempted execution of an MSR or MRS instruction using a [_EL12](#) register name when [HCR_EL2](#).E2H == 0.
- Attempted execution, in Debug state, of:
 - A DCPS1 instruction when the value of [HCR_EL2](#).TGE is 1 and EL2 is disabled or not implemented in the current Security state.
 - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
 - A DCPS3 instruction when the value of [EDSCR](#).SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using [R13_mon](#).
- In Debug state when the value of [EDSCR](#).SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to [SPSR_mon](#), [SP_mon](#), or [LR_mon](#).

- An exception that is taken to EL2 because the value of [HCR_EL2.TGE](#) is 1 that, if the value of [HCR_EL2.TGE](#) was 0 would have been reported with an ESR_ELx.EC value of 0b000111.
- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF* instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0										RN				RES0		RV	TI		

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:10]

Reserved, RES0.

RN, bits [9:5]**When FEAT_WFxT is implemented:**

Register Number. Indicates the register number supplied for a WFET or WFIT instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [4:3]

Reserved, RES0.

RV, bit [2]**When FEAT_WFxT is implemented:**

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped WFET or WFIT instruction.

RV	Meaning
0b0	Register field invalid.
0b1	Register field valid.

If TI[1] == 0, then this field is RES0.

This field is set to 1 on a trap on WFET or WFIT.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

TI	Meaning	Applies when
0b00	WFI trapped.	
0b01	WFE trapped.	
0b10	WFIT trapped.	When FEAT_WFxT is implemented
0b11	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating this exception:

- [SCTLR_EL1](#).{nTWE, nTWI}.
- [HCR_EL2](#).{TWE, TWI}.

- [SCR_EL3](#).{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc2			Opc1			CRn			Rt			CRm			Direction				

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCR instruction.
0b1	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#) or [CPACR](#) using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CNTHCTL_EL2](#).EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL3](#).TCPAC, for accesses to [CPACR](#) from EL1 and EL2, and accesses to [HCPTR](#) from EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- [CPACR_EL1](#).TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [HCR_EL2](#).TID0, for accesses to the [JIDR](#) register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

- [MDCR_EL3](#).TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == 0b1110) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001000:

- [HCR_EL2](#).TID0, for accesses to the [FPSID](#) register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- [HCR_EL2](#).TID3, for accesses to registers in ID group 3 including [MVFR0](#), [MVFR1](#) and [MVFR2](#), VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B* instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ISS											

ISS, bits [24:0]

ISS	Meaning	Applies when
0b000000000000000000000000000000	ST64BV instruction trapped.	When FEAT_LS64_V is implemented
0b000000000000000000000000000001	ST64BV0 instruction trapped.	When FEAT_LS64_ACCDATA is implemented
0b000000000000000000000000000010	LD64B or ST64B instruction trapped.	When FEAT_LS64 is implemented

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc1			RES0	Rt2				Rt			CRm			Direction					

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b1111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b1111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b11111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b11111.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCRR instruction.
0b1	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [AMUSERENR_EL0](#).{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [HSTR_EL2](#).T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL3.
- If FEAT_FGT is implemented, [HDFGRTR_EL2](#).PMCCNTR_EL0 for MRRC access and [HDFGWTR_EL2](#).PMCCNTR_EL0 for MCRR access to [PMCCNTR](#) at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- [MDSCR_EL1](#).TDCC, for accesses to the Debug ROM registers [DBGDSAR](#) and [DBGDRAR](#) at EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers [DBGDRAR](#) and [DBGDSAR](#) using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.
- [CPACR_EL1](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [CPTR_EL2](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL2.
- [CPTR_EL3](#).TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == 0b1110) trapped to EL3.

Note

If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are UNDEFINED and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				imm8								RES0			Rn				Offset		AM		Direction

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not 0b1111, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is 0b1111:

- If the instruction that generated the exception is not UNPREDICTABLE, then the register specifier takes the value 0b1111.
- If the instruction that generated the exception is UNPREDICTABLE, then the register specifier takes an UNKNOWN value, which is restricted to either:
 - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
 - The value 0b1111.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Offset	Meaning
0b0	Subtract offset.
0b1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
0b000	Immediate unindexed.
0b001	Immediate post-indexed.
0b010	Immediate offset.
0b011	Immediate pre-indexed.
0b100	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to memory. STC instruction.
0b1	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- [MDSCR_EL1](#).TDCC, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) trapped to EL1 or EL2.
- [MDCR_EL2](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL2.
- [MDCR_EL3](#).TDA, for accesses using AArch32 state, LDC access to [DBGDTRTXint](#) or STC access to [DBGDTRRXint](#) MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				RES0																			

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

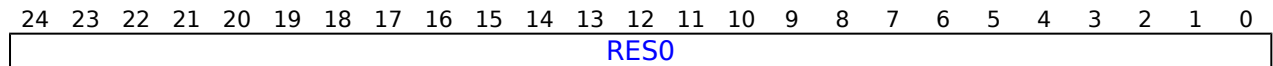
Bits [19:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000111:

- [CPACR_EL1.FPEN](#), for accesses to SIMD and floating-point registers trapped to EL1.
- [CPTR_EL2.FPEN](#) and [CPTR_EL2.TFP](#), for accesses to SIMD and floating-point registers trapped to EL2.
- [CPTR_EL3.TFP](#), for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ



The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value 0b000000.

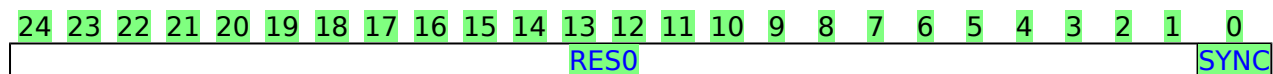
Bits [24:0]

Reserved, RES0.

The following fields describe the configuration settings for the traps that are reported using EC value 0b011001:

- [CPACR_EL1.ZEN](#), for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- [CPTR_EL2.ZEN](#) and [CPTR_EL2.TZ](#), for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- [CPTR_EL3.EZ](#), for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for a PMU exception



Bits [24:1]

Reserved, RES0.

SYNC, bit [0]

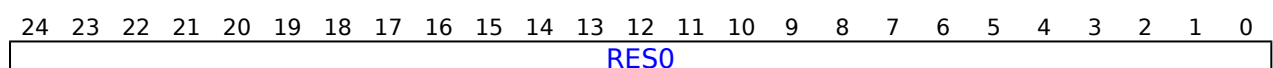
Indicates whether the exception was taken synchronously or asynchronously.

SYNC	Meaning	Applies when
0b0	The exception was taken asynchronously because an overflow status flag was set.	
0b1	The exception was taken synchronously because PSTATE.PPEND was set.	When FEAT_SEBEP is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault



Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from the Memory Copy and Memory Set instructions

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MemInst	isSETG	Options	FromEpilogue	WrongOption	OptionA	RES0	destreg	srcreg	sizereg															

MemInst, bit [24]

Indicates the memory instruction class causing the exception.

MemInst	Meaning
0b0	CPYFE*, CPYFM*, CPYE*, and CPYM* instructions.
0b1	SETE*, SETM*, SETGE*, and SETGM* instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

isSETG, bit [23]

Indicates whether the instruction belongs to SETGM* or SETGE* class of instruction.

isSETG	Meaning
0b0	Not a SETGM* or SETGE* instruction.
0b1	SETGM* or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Options, bits [22:19]

Options : the Options field of the instruction.

For Memory Copy instructions, bits[22:19] forms the Options field, which holds the bits[15:12] of the instruction.

For Memory Set instructions:

- Bits[22:21] are RES0.
- Bits[20:19] form the Options field, which holds the bits[13:12] of the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FromEpilogue, bit [18]

Indicates whether the instruction belongs to the epilogue class of Memory Copy or Memory Set instructions.

FromEpilogue	Meaning
0b0	Not an epilogue instruction.
0b1	CPYE*, CPYFE*, SETE*, or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WrongOption, bit [17]

Algorithm option.

WrongOption	Meaning
0b0	WrongOption is false.
0b1	WrongOption is true.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OptionA, bit [16]

Algorithm type indicated by the PSTATE.C bit.

OptionA	Meaning
0b0	OptionB indicated by PSTATE.C is 0.
0b1	OptionA indicated by PSTATE.C is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [15]

Reserved, RES0.

destreg, bits [14:10]

The destination register value from the issued instruction, containing the destination address.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

srcreg, bits [9:5]

The source register value from the issued instruction, containing either the source address or the source data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

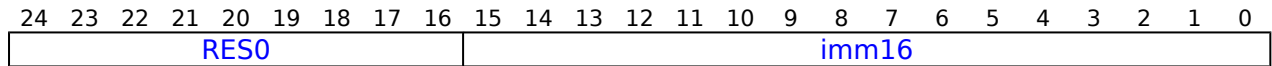
sizereg, bits [4:0]

The size register value from the issued instruction, containing the number of bytes to be transferred or set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from HVC or SVC instruction execution



Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
 - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
 - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

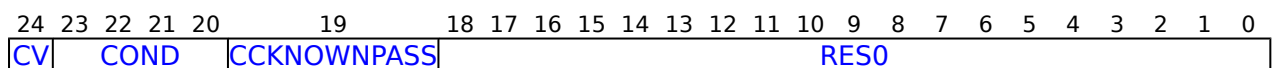
In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, [HFGITR_EL2](#).{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from SMC instruction execution in AArch32 state



For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from EL1 because [HCR_EL2](#).TSC is 1, the ISS encoding is as shown in the diagram.

CV, bit [24]

Condition code valid.

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to 0b1110.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
 - If the instruction is conditional, COND is set to the condition code field value from the instruction.
 - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
 - With COND set to 0b1110, the value for unconditional.
 - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
 - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
 - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is valid only if CCKNOWNPASS is 1, otherwise it is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

CCKNOWNPASS	Meaning
0b0	The instruction was unconditional, or was conditional and passed its condition code check.
0b1	The instruction was conditional, and might have failed its condition code check.

Note

In an implementation in which an SMC instruction that fails its code check is not trapped, this field can always return the value 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [18:0]

Reserved, RES0.

[HCR_EL2](#).TSC describes the configuration settings for trapping SMC instructions to EL2.

ISS encoding for an exception from SMC instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										imm16														

Bits [24:16]

Reserved, RES0.

imm16, bits [15:0]

The value of the immediate field from the issued SMC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The value of ISS[24:0] described here is used both:

- When an SMC instruction is trapped from EL1 modes.
- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

[HCR_EL2](#).TSC describes the configuration settings for trapping SMC from EL1 modes.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				Op0		Op2		Op1		CRn				Rt				CRm			Direction			

Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write access, including MSR instructions.
0b1	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:

- If FEAT_TIDCP1 is implemented, [SCTLR_EL1](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL1.
- [SCTLR_EL1](#).UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UCT, for accesses to [CTR_EL0](#) using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [SCTLR_EL1](#).UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [CPACR_EL1](#).TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, [MDCR_EL2](#).TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and [MDCR_EL3](#).TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [PMUSERENR_EL0](#).{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.

- [AMUSERENR_EL0](#).EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- [HCR_EL2](#).{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TACR, for accesses to the Auxiliary Control Register, [ACTLR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- If FEAT_TIDCP1 is implemented, [SCTLR_EL2](#).TIDCP, for EL0 accesses to IMPLEMENTATION DEFINED functionality using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TCPAC, for accesses to [CPACR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TTRF, for accesses to the trace filter control register, [TRFCR_EL1](#), using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [CNTHCTL_EL2](#).{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- [MDCR_EL2](#).{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [CPTR_EL2](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- [HCR_EL2](#).{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- [HCR_EL2](#).AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- [HCR_EL2](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.
- [SCR_EL3](#).APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [SCR_EL3](#).{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TCPAC, for accesses to [CPTR_EL2](#) and [CPACR_EL1](#) using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TTRF, for accesses to the trace filter control registers, [TRFCR_EL1](#) and [TRFCR_EL2](#), using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [MDCR_EL3](#).TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- [CPTR_EL3](#).TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
 - [HCR_EL2](#).{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
 - [HCR2](#).{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:

- [SCR_EL3.FGTEn](#), for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
- [HFGTR_EL2](#) for reads and [HFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
- [HFGITR_EL2](#) for execution of system instructions, MSR or MRS access trapped to EL2
- [HDFGTR_EL2](#) for reads and [HDFGWTR_EL2](#) for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
- [HAFGRTR_EL2](#) for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
 - [SCR_EL3.TRNDR](#) for reads of [RNDR](#) and [RNDRRS](#) using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
 - [CPTR_EL3.ESM](#), for MSR or MRS accesses to [SMPRI_EL1](#) at EL1, EL2, and EL3, trapped to EL3.
 - [CPTR_EL3.ESM](#), for MSR or MRS accesses to [SMPRMAP_EL2](#) at EL2 and EL3, trapped to EL3.
 - [SCTLR_EL1.EnTP2](#), for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL1 or EL2.
 - [SCTLR_EL2.EnTP2](#), for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, trapped to EL2.
 - [SCR_EL3.EnTP2](#), for MSR or MRS accesses to [TPIDR2_EL0](#) at EL0, EL1, and EL2, trapped to EL3.
- If FEAT_NMI is implemented, [HCRX_EL2.TALLINT](#), for MSR writes of [ALLINT](#) at EL1, trapped to EL2.

ISS encoding for an exception from MSRR, MRRS, or 128-bit System instruction execution in AArch64 state

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RES0				Op0			Op2			Op1			CRn			Rt			RES0			CRm			Direction	

Bits [24:22]

Reserved, RES0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Rt, bits [9:6]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

Note

This value represents register pair of X[Rt:0], X[Rt:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write access, MSRR instructions.
0b1	Read access, MRRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an IMPLEMENTATION DEFINED exception to EL3

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPLEMENTATION DEFINED																								

IMPLEMENTATION DEFINED, bits [24:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from an Instruction Abort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												SET	FnV	EA	RES0	S1PTW	RES0	IFSC						

When FEAT_S1POE or FEAT_S2POE is implemented, if a memory access generates a Instruction Abort due to a Permission fault, the ISS2 encoding for an exception from an Instruction Abort includes further information about the exception.

Bits [24:13]

Reserved, RES0.

SET, bits [12:11]**When FEAT_RAS is implemented:**

Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the IFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [8]

Reserved, RES0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or	When FEAT_RAS is not implemented

0b011110	hardware update of translation table, level 1. Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The **lookup** level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception due to SME functionality

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																						SMTC		

The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#).

Bits [24:3]

Reserved, RES0.

SMTC, bits [2:0]

SME Trap Code. Identifies the reason for instruction trapping.

SMTC	Meaning	Applies when
0b000	Access to SME functionality trapped as a result of CPACR_EL1 .SMEN, CPTR_EL2 .SMEN, CPTR_EL2 .TSM, or CPTR_EL3 .ESM, that is not reported using EC 0b000000.	
0b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.	
0b010	SME instruction trapped because PSTATE.SM is 0.	
0b011	SME instruction trapped because PSTATE.ZA is 0.	
0b100	Access to the SME2 ZT0 register trapped as a result of SMCR_EL1 .EZT0, SMCR_EL2 .EZT0, or SMCR_EL3 .EZT0.	When FEAT_SME2 is implemented

All other values are reserved.

The following fields describe the configuration settings for the traps that are reported using the EC value 0b011101:

- [CPACR_EL1](#).SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVCR](#) and [SMCR_EL1](#) System registers at EL1 and EL0, trapped to EL1 or EL2.
- [CPTR_EL2](#).SMEN and [CPTR_EL2](#).TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#) at EL2, EL1, or EL0, trapped to EL2.
- [CPTR_EL3](#).ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access [SVCR](#), [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#) from all Exception levels and any Security state, trapped to EL3.
- If FEAT_SME2 is implemented:
 - [SMCR_EL1](#).EZT0, for accesses to ZT0 at EL1 and EL0, trapped to EL1 or EL2.
 - [SMCR_EL2](#).EZT0, for accesses to ZT0 at EL2, EL1, and EL0, trapped to EL2.
 - [SMCR_EL3](#).EZT0, for accesses to ZT0 at any Exception level, trapped to EL3.

ISS encoding for an exception from a Granule Protection Check

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0		S2PTWInD		GPCSC				VNCR		RES0		CMS1PTW		WnR		xFSC								

Bits [24:22]

Reserved, RES0.

S2PTW, bit [21]

Indicates whether the Granule Protection Check exception was on an access made for a stage 2 translation table walk.

S2PTW	Meaning
0b0	Fault not on a stage 2 translation table walk.
0b1	Fault on a stage 2 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

InD, bit [20]

Indicates whether the Granule Protection Check exception was on an instruction or data access.

InD	Meaning
0b0	Data access.
0b1	Instruction access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPCSC, bits [19:14]

Granule Protection Check Status Code.

GPCSC	Meaning
0b000000	GPT address size fault at level 0.
0b000100	GPT walk fault at level 0.
0b000101	GPT walk fault at level 1.
0b001100	Granule protection fault at level 0.
0b001101	Granule protection fault at level 1.
0b010100	Synchronous External abort on GPT fetch at level 0.
0b010101	Synchronous External abort on GPT fetch at level 1.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

When InD is '1', this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

Indicates whether the Granule Protection Check exception was on an access for stage 2 translation for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

When InD is '1', this field is RES0.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

xFSC, bits [5:0]

Instruction or Data Fault Status Code.

xFSC	Meaning	Applies when
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The **lookup** level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a Data Abort

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE	SRT			Bit[15]			AR	VNCR	Bits[12:11]			FnV	EA	CM	S1PTW	WnR	DFSC					

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, **then** this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, **then** this ISS encoding includes ISS2, bits[36:32].

When FEAT_S1POE is implemented, if a memory access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

When FEAT_S1PIE is implemented, if a memory write access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0b0	No valid instruction syndrome. ISS[23:14] are RES0.
0b1	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
 - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
 - Is not performing register writeback.
 - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FnV is 0 and FAR_ELx is valid.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SAS, bits [23:22]

When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0b11.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSE, bit [21]

When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

SSE	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SRT, bits [20:16]

When ISV == 1:

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[15]

When ISV == 1:

SF, bit [0] of bit [15]

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
0b0	Instruction loads/stores a 32-bit general-purpose register.
0b1	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When ISV == 0 and FEAT_SME is implemented:

FnP, bit [0] of bit [15]

FAR not Precise.

FnP	Meaning	Applies when
0b0	The FAR holds the faulting virtual address that generated the Data Abort.	
0b1	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction in Streaming SVE mode, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).	When FEAT_SME is implemented or FEAT_SVE is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AR, bit [14]

When ISV == 1:

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VNCR, bit [13]**When FEAT_NV2 is implemented:**

Indicates that the fault came from use of [VNCR_EL2](#) register by EL1 code.

VNCR	Meaning
0b0	The fault was not generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.
0b1	The fault was generated by the use of VNCR_EL2 , by an MRS or MSR instruction executed at EL1.

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits[12:11]**When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:****LST, bits [1:0] of bits [12:11]**

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	
0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RAS is implemented and DFSC == 0b010000:**SET, bits [1:0] of bits [12:11]**

Synchronous Error Type. Used when a Synchronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning
0b00	Recoverable state (UER).
0b10	Uncontainable (UC).
0b11	Restartable state (UEO).

All other values are reserved.

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	FAR is valid.
0b1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 0b010000. It is RES0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0b0	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0b0	Fault not on a stage 2 translation for a stage 1 translation table walk.
0b1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Abort caused by an instruction reading from a memory location.
0b1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An External abort on an Atomic access.
- A fault reported using a DFSC value of 0b110101 or 0b110001, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented

ESR_EL3, Exception Syndrome Register (EL3)

0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The **lookup** level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ISS encoding for an exception from a trapped floating-point exception

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	TFV							RES0						VECITR	IDF	RES0	IXF	UFF	OFF	DZF	IOF			

Bit [24]

Reserved, RES0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

TFV	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

Note

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [22:11]

Reserved, RES0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IDF	Meaning
0b0	Input denormal floating-point exception has not occurred.
0b1	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IXF	Meaning
0b0	Inexact floating-point exception has not occurred.
0b1	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

UFF	Meaning
0b0	Underflow floating-point exception has not occurred.
0b1	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

OFF	Meaning
0b0	Overflow floating-point exception has not occurred.
0b1	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

DZF	Meaning
0b0	Divide by Zero floating-point exception has not occurred.
0b1	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IOF	Meaning
0b0	Invalid Operation floating-point exception has not occurred.
0b1	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the [FPCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the [FPSCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for an SError interrupt

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDS	RES0										IESB	AET		EA	RES0		DFSC							

IDS, bit [24]

IMPLEMENTATION DEFINED syndrome.

IDS	Meaning
0b0	Bits [23:0] of the ISS field holds the fields described in this encoding.
Note If FEAT_RAS is not implemented, bits [23:0] of the ISS field are RES0.	
0b1	Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

Note

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [23:14]

Reserved, RES0.

IESB, bit [13]**When FEAT_IESB is implemented and DFSC == 0b010001:**

Implicit error synchronization event.

IESB	Meaning
0b0	The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately.
0b1	The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AET, bits [12:10]**When FEAT_RAS is implemented and DFSC == 0b010001:**

Asynchronous Error Type.

Describes the PE error state after taking the SError interrupt exception.

AET	Meaning
0b000	Uncontainable (UC).
0b001	Unrecoverable state (UEU).
0b010	Restartable state (UEO).
0b011	Recoverable state (UER).
0b110	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Note

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA, bit [9]**When FEAT_RAS is implemented and DFSC == 0b010001:**

External abort type. Provides an IMPLEMENTATION DEFINED classification of External aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [8:6]

Reserved, RES0.

DFSC, bits [5:0]**When FEAT_RAS is implemented:**

Data Fault Status Code.

DFSC	Meaning
0b000000	Uncategorized error.
0b010001	Asynchronous SError interrupt.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ISS encoding for an exception from execution of a Breakpoint instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										Comment														

Bits [24:16]

Reserved, RES0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from a TSTART instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0															Rd					RES0				

Bits [24:10]

Reserved, RES0.

Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, RES0.

ISS encoding for an exception from Branch Target Identification instruction

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																						BTTYPE		

Bits [24:2]

Reserved, RES0.

BTTYPE, bits [1:0]

This field is set to the PSTATE.BTTYPE value that generated the Branch Target Exception.

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 || SCR_EL3.API == 0

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								

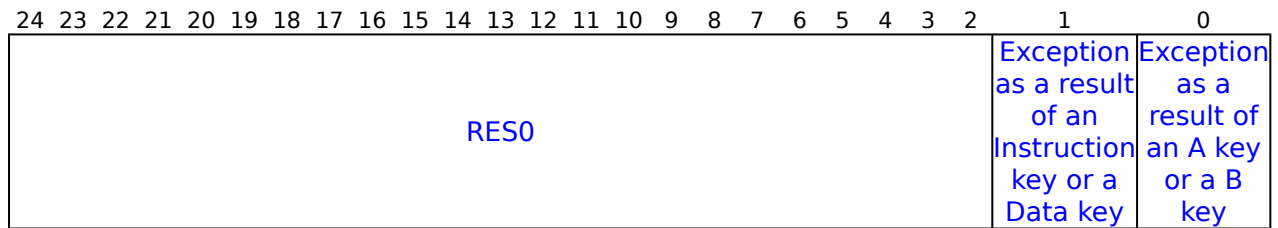
Bits [24:0]

Reserved, RES0.

For more information about generating these exceptions, see:

- [HCR_EL2.API](#), for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2.
- [SCR_EL3.API](#), for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL3.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure



Bits [24:2]

Reserved, RES0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

Meaning	
0b0	Instruction Key.
0b1	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

Meaning	
0b0	A key.
0b1	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- AUTIASP, AUTIAZ, AUTIA1716.
- AUTIBSP, AUTIBZ, AUTIB1716.
- AUTIA, AUTDA, AUTIB, AUTDB.
- AUTIZA, AUTIZB, AUTDZA, AUTDZB.

It is IMPLEMENTATION DEFINED whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- RETAA, RETAB.
- BRAA, BRAB, BLRAA, BLRAB.
- BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB.
- LDRAA, LDRAB, whether the authenticated address is written back to the base register or not.

Accessing ESR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ESR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL3;
```

MSR ESR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ESR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

FAR_EL1, Fault Address Register (EL1)

The FAR_EL1 characteristics are:

Purpose

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions, PC alignment fault exceptions and Watchpoint exceptions that are taken to EL1.

Configuration

AArch64 System register FAR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DFAR\[31:0\]](#) (DFAR_NS).

AArch64 System register FAR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [IFAR\[31:0\]](#) (IFAR_NS).

Attributes

FAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Faulting Virtual Address for synchronous exceptions taken to EL1																															
Faulting Virtual Address for synchronous exceptions taken to EL1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL1. Exceptions that set the FAR_EL1 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). [ESR_EL1](#).EC holds the EC value for the exception.

For a synchronous External abort, if the VA that generated the abort was from an address range for which TCR_ELx.TBI{<0|1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL1 are UNKNOWN.

For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if [ESR_EL1](#).FnV is 0, and FAR_EL1 is UNKNOWN if [ESR_EL1](#).FnV is 1.

If a memory fault that sets FAR_EL1, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.

On an exception due to a Tag Check Fault caused by a data cache maintenance or other DC instruction, the address held in FAR_EL1 is IMPLEMENTATION DEFINED as one of the following:

- The lowest address that gave rise to the fault.
- The address specified in the register argument of the instruction as generated by MMU faults caused by [DC ZVA](#).

If the exception that updates FAR_EL1 is taken from an Exception level using AArch32, the top 32 bits are all zero, unless both of the following apply, in which case the top 32 bits of FAR_ELx are 0x00000001:

- The faulting address was generated by a load or store instruction that sequentially incremented from address 0xFFFFFFFF. Such a load or store instruction is CONSTRAINED UNPREDICTABLE.
- The implementation treats such incrementing as setting bit[32] of the virtual address to 1.

When the PE sets [ESR_EL1.{ISV,FnP}](#) to {0,1} on taking a Data Abort exception, or sets [ESR_EL1.{FnV,FnP}](#) to {0,1} on taking a Watchpoint exception, the PE sets FAR_EL1 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort exception or Watchpoint exception.

~~When FEAT_SME is implemented, and the PE sets [ESR_EL1.ISV](#) to 0 and [ESR_EL1.FnP](#) to 1 on taking a Data Abort exception or Watchpoint exception, the PE sets FAR_EL1 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort or Watchpoint exception.~~

The naturally-aligned fault granule is one of:

- When ESR_EL1.DFSC is 0b010001, indicating a Synchronous Tag Check fault, it is a 16-byte tag granule.
- When ESR_EL1.DFSC is 0b11010x, indicating an IMPLEMENTATION DEFINED fault, it is an IMPLEMENTATION DEFINED granule.
- Otherwise, it is the smallest implemented translation granule.

When FEAT_MOPS is implemented, the value in FAR_EL1 on a synchronous exception from any of the Memory Copy and Memory Set instructions represents the first element that has not been copied or set, and is determined as follows:

- For a Data Abort generated by the MMU, the value is within the address range of the relevant translation granule, aligned to the size of the relevant translation granule of the address that generated the Data Abort. Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the relevant translation granule size in bytes. For the purpose of calculating the relevant translation granule, if the MMU is disabled for a stage of translation, then the current translation granule size is equal to 2^{64} for stage 1, and the PARange for stage 2. The relevant translation granule is:
 - For MMU faults generated at stage 1, the current stage 1 translation granule.
 - For MMU faults generated at stage 2, the smaller of the current stage 1 translation granule and the current stage 2 translation granule.
 - If FEAT_RME is implemented, for a synchronous data abort generated as the result of a GPF, the smallest of the current stage 1 translation granule, the current stage 2 translation granule and the configured granule size in [GPCCR_EL3.PGS](#).
- For a Data Abort generated by a Tag Check failure, the value is the lowest address that failed the Tag Check within the block size of the load or store.
- For a Watchpoint exception, the value is an address range of the size defined by the [DCZID_ELO.BS](#) field. This address does not need to be the element with a watchpoint, but can be some earlier element.
- Otherwise, the value is the lowest address in the block size of the load or store.

For a Data Abort [exception](#) or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging [in AArch64 state](#)'.

[When FEAT_MTE_TAGGED_FAR is not implemented, on](#)~~For~~ a synchronous Tag Check Fault abort, bits[63:60] are UNKNOWN.

Execution at EL0 makes FAR_EL1 become UNKNOWN.

Note

The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lower address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

For all other exceptions taken to EL1, FAR_EL1 is UNKNOWN.

FAR_EL1 is made UNKNOWN on an exception return from EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing FAR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic FAR_EL1 or FAR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, FAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.FAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x220];
    else
        X[t, 64] = FAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = FAR_EL2;
    else
        X[t, 64] = FAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL1;

```

MSR FAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.FAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x220] = X[t, 64];
    else
        FAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        FAR_EL2 = X[t, 64];
    else
        FAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    FAR_EL1 = X[t, 64];

```

MRS <Xt>, FAR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x220];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = FAR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = FAR_EL1;
    else
        UNDEFINED;

```

MSR FAR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x220] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        FAR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        FAR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

MRS <Xt>, FAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = FAR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = FAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL2;

```

MSR FAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        FAR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    FAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    FAR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

FAR_EL2, Fault Address Register (EL2)

The FAR_EL2 characteristics are:

Purpose

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions, PC alignment fault exceptions and Watchpoint exceptions that are taken to EL2.

Configuration

AArch64 System register FAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HDFAR\[31:0\]](#).

AArch64 System register FAR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HIFAR\[31:0\]](#).

AArch64 System register FAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [DFAR\[31:0\]](#) (DFAR_S) when EL2 is implemented.

AArch64 System register FAR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [IFAR\[31:0\]](#) (IFAR_S) when EL2 is implemented.

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

FAR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Faulting Virtual Address for synchronous exceptions taken to EL2																															
Faulting Virtual Address for synchronous exceptions taken to EL2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL2. Exceptions that set the FAR_EL2 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). [ESR_EL2](#).EC holds the EC value for the exception.

For a synchronous External abort, if the VA that generated the abort was from an address range for which TCR_ELx.TBI{<0|1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL2 are UNKNOWN.

For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if [ESR_EL2](#).FnV is 0, and FAR_EL2 is UNKNOWN if [ESR_EL2](#).FnV is 1.

If a memory fault that sets FAR_EL2, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.

On an exception due to a Tag Check Fault caused by a data cache maintenance or other DC instruction, the address held in FAR_EL2 is IMPLEMENTATION DEFINED as one of the following:

- The lowest address that gave rise to the fault.
- The address specified in the register argument of the instruction as generated by MMU faults caused by [DC ZVA](#).

If the exception that updates FAR_EL2 is taken from an Exception level using AArch32, the top 32 bits are all zero, unless both of the following apply, in which case the top 32 bits of FAR_ELx are 0x00000001:

- The faulting address was generated by a load or store instruction that sequentially incremented from address 0xFFFFFFFF. Such a load or store instruction is CONSTRAINED UNPREDICTABLE.
- The implementation treats such incrementing as setting bit[32] of the virtual address to 1.

When the PE sets ESR_EL2.{ISV,FnP} to {0,1} on taking a Data Abort exception, or sets ESR_EL2.{FnV,FnP} to {0,1} on taking a Watchpoint exception, the PE sets FAR_EL2 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort exception or Watchpoint exception.

When FEAT_SME is implemented, and the PE sets ESR_EL2.ISV to 0 and ESR_EL2.FnP to 1 on taking a Data Abort exception or Watchpoint exception, the PE sets FAR_EL2 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort or Watchpoint exception.

The naturally-aligned fault granule is one of:

- When ESR_EL2.DFSC is 0b010001, indicating a Synchronous Tag Check fault, it is a 16-byte tag granule.
- When ESR_EL2.DFSC is 0b11010x, indicating an IMPLEMENTATION DEFINED fault, it is an IMPLEMENTATION DEFINED granule.
- Otherwise, it is the smallest implemented translation granule.

When FEAT_MOPS is implemented, the value in FAR_EL2 on a synchronous exception from any of the Memory Copy and Memory Set instructions represents the first element that has not been copied or set, and is determined as follows:

- For a Data Abort generated by the MMU, the value is within the address range of the relevant translation granule, aligned to the size of the relevant translation granule of the address that generated the Data Abort. Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the relevant translation granule size in bytes. For the purpose of calculating the relevant translation granule, if the MMU is disabled for a stage of translation, then the current translation granule size is equal to 2^{64} for stage 1, and the PARange for stage 2. The relevant translation granule is:
 - For MMU faults generated at stage 1, the current stage 1 translation granule.
 - For MMU faults generated at stage 2, the smaller of the current stage 1 translation granule and the current stage 2 translation granule.
 - If FEAT_RME is implemented, for a synchronous data abort generated as the result of a GPF, the smallest of the current stage 1 translation granule, the current stage 2 translation granule and the configured granule size in GPCCR_EL3.PGS.
- For a Data Abort generated by a Tag Check failure, the value is the lowest address that failed the Tag Check within the block size of the load or store.
- For a Watchpoint exception, the value is an address range of the size defined by the DCZID_ELO.BS field. This address does not need to be the element with a watchpoint, but can be some earlier element.
- Otherwise, the value is the lowest address in the block size of the load or store.

For a Data Abort exception or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging in AArch64 state'.

When FEAT_MTE_TAGGED_FAR is not implemented, on For a synchronous Tag Check Fault abort, bits[63:60] are UNKNOWN.

Execution at EL1 or EL0 makes FAR_EL2 become UNKNOWN.

Note

The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lower address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

For all other exceptions taken to EL2, FAR_EL2 is UNKNOWN.

FAR_EL2 is made UNKNOWN on an exception return from EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing FAR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic FAR_EL2 or FAR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, FAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = FAR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = FAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL2;

```

MSR FAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        FAR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    FAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    FAR_EL2 = X[t, 64];

```

MRS <Xt>, FAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.FAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x220];
    else
        X[t, 64] = FAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = FAR_EL2;
    else
        X[t, 64] = FAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL1;

```

MSR FAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.FAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x220] = X[t, 64];
    else
        FAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        FAR_EL2 = X[t, 64];
    else
        FAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    FAR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

FAR_EL3, Fault Address Register (EL3)

The FAR_EL3 characteristics are:

Purpose

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions and PC alignment fault exceptions that are taken to EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to FAR_EL3 are UNDEFINED.

Attributes

FAR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Faulting Virtual Address for synchronous exceptions taken to EL3																															
Faulting Virtual Address for synchronous exceptions taken to EL3																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL3. Exceptions that set the FAR_EL3 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), and PC alignment faults (EC 0x22). [ESR_EL3](#).EC holds the EC value for the exception.

For a synchronous External abort, if the VA that generated the abort was from an address range for which TCR_ELx.TBI{<0|1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL3 are UNKNOWN.

For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if [ESR_EL3](#).FnV is 0, and FAR_EL3 is UNKNOWN if [ESR_EL3](#).FnV is 1.

If a memory fault that sets FAR_EL3, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.

On an exception due to a Tag Check Fault caused by a data cache maintenance or other DC instruction, the address held in FAR_EL3 is IMPLEMENTATION DEFINED as one of the following:

- The lowest address that gave rise to the fault.
- The address specified in the register argument of the instruction as generated by MMU faults caused by [DC ZVA](#).

If the exception that updates FAR_EL3 is taken from an Exception level using AArch32, the top 32 bits are all zero, unless both of the following apply, in which case the top 32 bits of FAR_ELx are 0x00000001:

- The faulting address was generated by a load or store instruction that sequentially incremented from address 0xFFFFFFFF. Such a load or store instruction is CONSTRAINED UNPREDICTABLE.
- The implementation treats such incrementing as setting bit[32] of the virtual address to 1.

When the PE sets [ESR_EL3](#).{ISV,FnP} to {0,1} on taking a Data Abort exception, the PE sets FAR_EL3 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort exception.

When FEAT_SME is implemented, and the PE sets ESR_EL3.ISV to 0 and ESR_EL3.FnP to 1 on taking a Data Abort exception, the PE sets FAR_EL3 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort.

The naturally-aligned fault granule is one of:

- When ESR_EL3.DFSC is 0b010001, indicating a Synchronous Tag Check fault, it is a 16-byte tag granule.
- When ESR_EL3.DFSC is 0b11010x, indicating an IMPLEMENTATION DEFINED fault, it is an IMPLEMENTATION DEFINED granule.
- Otherwise, it is the smallest implemented translation granule.

When FEAT_MOPS is implemented, the value in FAR_EL3 on a synchronous exception from any of the Memory Copy and Memory Set instructions represents the first element that has not been copied or set, and is determined as follows:

- For a Data Abort generated by the MMU, the value is within the address range of the relevant translation granule, aligned to the size of the relevant translation granule of the address that generated the Data Abort. Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the relevant translation granule size in bytes. For the purpose of calculating the relevant translation granule, if the MMU is disabled for a stage of translation, then the current translation granule size is equal to 2^{64} for stage 1, and the PARange for stage 2. The relevant translation granule is:
 - For MMU faults generated at stage 1, the current stage 1 translation granule.
 - For MMU faults generated at stage 2, the smaller of the current stage 1 translation granule and the current stage 2 translation granule.
 - If FEAT_RME is implemented, for a synchronous data abort generated as the result of a GPF, the smallest of the current stage 1 translation granule, the current stage 2 translation granule and the configured granule size in GPCCR_EL3.PGS.
- For a Data Abort generated by a Tag Check failure, the value is the lowest address that failed the Tag Check within the block size of the load or store.
- Otherwise, the value is the lowest address in the block size of the load or store.

For a Data Abort exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging in AArch64 state'.

When FEAT_MTE_TAGGED_FAR is not implemented, on For a synchronous Tag Check Fault abort, bits[63:60] are UNKNOWN.

Execution at EL2, EL1, or EL0 makes FAR_EL3 become UNKNOWN.

Note

The address held in this register is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lowest address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

For all other exceptions taken to EL3, FAR_EL3 is UNKNOWN.

FAR_EL3 is made UNKNOWN on an exception return from EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing FAR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, FAR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL3;
```

MSR FAR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    FAR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

FPCR, Floating-point Control Register

The FPCR characteristics are:

Purpose

Controls floating-point behavior.

Configuration

AArch64 System register FPCR bits [26:15] are architecturally mapped to AArch32 System register [FPSCR\[26:15\]](#).

AArch64 System register FPCR bits [12:8] are architecturally mapped to AArch32 System register [FPSCR\[12:8\]](#).

It is IMPLEMENTATION DEFINED whether the Len and Stride fields can be programmed to non-zero values, which will cause some AArch32 floating-point instruction encodings to be UNDEFINED, or whether these fields are RAZ.

Attributes

FPCR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32																																																
																RES0																																																															
RES0				AHP				DN				FZ				RMode				Stride				FZ16				Len				IDE				RES0				EBF				IXE				UFE				OFE				DZE				IOE				RES0				NEP				AH				FIZ			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																

Bits [63:27]

Reserved, RES0.

AHP, bit [26]

Alternative half-precision control bit.

AHP	Meaning
0b0	IEEE half-precision format selected.
0b1	Alternative half-precision format selected.

This bit is used only for conversions between half-precision floating-point and other floating-point formats.

The data-processing instructions added as part of the FEAT_FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DN, bit [25]

Default NaN use for NaN propagation.

DN	Meaning
0b0	NaN operands propagate through to the output of a floating-point operation.
0b1	Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of FABS, FMAX*, FMIN*, and FNEG instructions, and a default NaN is never returned as a result of these instructions.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FZ, bit [24]

Flushing denormalized numbers to zero control bit.

FZ	Meaning
0b0	If FPCR.AH is 0, the flushing to zero of single-precision and double-precision denormalized inputs to, and outputs of, floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. If FPCR.AH is 1, the flushing to zero of single-precision and double-precision denormalized outputs of floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero.
0b1	If FPCR.AH is 0, denormalized single-precision and double-precision inputs to, and outputs from, floating-point instructions are flushed to zero. If FPCR.AH is 1, denormalized single-precision and double-precision outputs from floating-point instructions are flushed to zero.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

RMode, bits [23:22]

Rounding Mode control field.

RMode	Meaning
0b00	Round to Nearest (RN) mode.
0b01	Round towards Plus Infinity (RP) mode.
0b10	Round towards Minus Infinity (RM) mode.
0b11	Round towards Zero (RZ) mode.

The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.

If FPCR.AH is 1, then the following instructions use Round to Nearest mode regardless of the value of this bit:

- The FRECPPE, FRECPSP, FRECPXP, FRSQRTE, and FRSQRSTS instructions.
- The BFCVT, BFCVTN, BFCVTN2, BFCVTNT, BFMLALB, and BFMLALT instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Stride, bits [21:20]

This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state.

This field is included only for context saving and restoration of the AArch32 [FPSCR](#).Stride field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FZ16, bit [19]**When FEAT_FP16 is implemented:**

Flushing denormalized numbers to zero control bit on half-precision data-processing instructions.

FZ16	Meaning
0b0	For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers.
0b1	Flushing denormalized numbers to zero enabled. For some instructions that do not convert a half-precision input to a higher precision output, this bit enables flushing to zero of inputs and outputs that are half-precision denormalized numbers.

The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Len, bits [18:16]

This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state.

This field is included only for context saving and restoration of the AArch32 [FPSCR](#).Len field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IDE, bit [15]

Input Denormal floating-point exception trap enable.

IDE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR .IDC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR .IDC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.IDE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [14]

Reserved, RES0.

EBF, bit [13]

When FEAT_EBF16 is implemented:

The value of this bit controls the numeric behaviors of BFloat16 dot product calculations performed by the BFDOT, BFMLLA, BFMOPA, and BFMOPS instructions. **If FEAT_SME2 is implemented, this also controls BFVDOT instruction.**

When [ID_AA64ISAR1_EL1](#).BF16 and [ID_AA64ZFR0_EL1](#).BF16 are 0b0010, the PE supports the FPCR.EBF field. Otherwise, FPCR.EBF is RES0.

EBF	Meaning
0b0	These instructions use the standard BFloat16 behaviors: <ul style="list-style-type: none"> • Ignoring the FPCR.RMode control and using the rounding mode defined for BFloat16. For more information, see 'Round to Odd mode'. • Flushing denormalized inputs and outputs to zero, as if the FPCR.FZ and FPCR.FIZ controls had the value '1'. • Performing unfused multiplies and additions with intermediate rounding of all products and sums.
0b1	These instructions use the extended BFloat16 behaviors: <ul style="list-style-type: none"> • Supporting all four IEEE 754 rounding modes selected by the FPCR.RMode control. • Optionally, flushing denormalized inputs and outputs to zero, as governed by the FPCR.FZ and FPCR.FIZ controls. • Performing a fused two-way sum-of-products for each pair of adjacent BFloat16 elements, without intermediate rounding of the products, but rounding the single-precision sum before addition to the accumulator. • Generating the default NaN as intermediate sum-of-products when any multiplier input is a NaN, or any product is infinity \times 0.0, or there are infinite products with differing signs. • Generating an intermediate sum-of-products of the same infinity when there are infinite products all with the same sign.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

IXE, bit [12]

Inexact floating-point exception trap enable.

IXE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR .IXC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR .IXC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.IXE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

UFE, bit [11]

Underflow floating-point exception trap enable.

UFE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.UFC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs and Flush-to-zero is not enabled, the PE does not update the FPSR.UFC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.UFE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

OFE, bit [10]

Overflow floating-point exception trap enable.

OFE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.OFC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.OFC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.OFE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DZE, bit [9]

Divide by Zero floating-point exception trap enable.

DZE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.DZC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.DZC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.DZE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IOE, bit [8]

Invalid Operation floating-point exception trap enable.

IOE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR .IOC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR .IOC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.IOE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The **Effective** value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:3]

Reserved, RES0.

**NEP, bit [2]
When FEAT_AFP is implemented:**

Controls how the output elements other than the lowest element of the vector are determined for Advanced SIMD scalar instructions.

NEP	Meaning
0b0	Does not affect how the output elements other than the lowest are determined for Advanced SIMD scalar instructions.
0b1	<p>The output elements other than the lowest are taken from the following registers:</p> <ul style="list-style-type: none"> For 3-input scalar versions of the FMLA (by element) and FMLS (by element) instructions, the <Hd>, <Sd>, or <Dd> register. For 3-input versions of the FMADD, FMSUB, FNMADD, and FNMSUB instructions, the <Ha>, <Sa>, or <Da> register. For 2-input scalar versions of the FACGE, FACGT, FCMEQ (register), FCMGE (register), and FCMGT (register) instructions, the <Hm>, <Sm>, or <Dm> register. For 2-input scalar versions of the FABD, FADD (scalar), FDIV (scalar), FMAX (scalar), FMAXNM (scalar), FMIN (scalar), FMINNM (scalar), FMUL (by element), FMUL (scalar), FMULX (by element), FMULX, FNMUL (scalar), FRECPS, FRSQRTS, and FSUB (scalar) instructions, the <Hn>, <Sn>, or <Dn> register. For 1-input scalar versions of the following instructions, the <Hd>, <Sd>, or <Dd> register: <ul style="list-style-type: none"> The (vector) versions of the FCVTAS, FCVTAU, FCVTMS, FCVTMU, FCVTNS, FCVTNU, FCVTPS, and FCVTPU instructions. The (vector, fixed-point) and (vector, integer) versions of the FCVTZS, FCVTZU, SCVTF, and UCVTF instructions. The (scalar) versions of the FABS, FNEG, FRINT32X, FRINT32Z, FRINT64X, FRINT64Z, FRINTA, FRINTI, FRINTM, FRINTN, FRINTP, FRINTX, FRINTZ, and FSQRT instructions. The (scalar, fixed-point) and (scalar, integer) versions of the SCVTF and UCVTF instructions. The BFCVT, FCVT, FCVTXN, FRECPE, FRECPX, and FRSQRTE instructions.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled at the current Exception level, the value of FPCR.NEP is treated as 0 for all purposes other than a direct read or write of the FPCR.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AH, bit [1]

When FEAT_AFP is implemented:

Alternate Handling. Controls alternate handling of floating-point numbers.

The Arm architecture supports two models for handling some of the corner cases of the floating-point behaviors, such as the nature of flushing of denormalized numbers, the detection of tininess and other exceptions and a range of other behaviors. The value of the FPCR.AH bit selects between these models.

For more information on the FPCR.AH bit, see 'Flushing denormalized numbers to zero', 'Floating-point exceptions and exception traps' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FIZ, bit [0]**When FEAT_AFP is implemented:**

Flush Inputs to Zero. Controls whether single-precision, double-precision and BFloat16 input operands that are denormalized numbers are flushed to zero.

FIZ	Meaning
0b0	The flushing to zero of single-precision and double-precision denormalized inputs to floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero.
0b1	Denormalized single-precision and double-precision inputs to most floating-point instructions flushed to zero.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing FPCR

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, FPCR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;

```

MSR FPCR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GPCCR_EL3, Granule Protection Check Control Register (EL3)

The GPCCR_EL3 characteristics are:

Purpose

The control register for Granule Protection Checks.

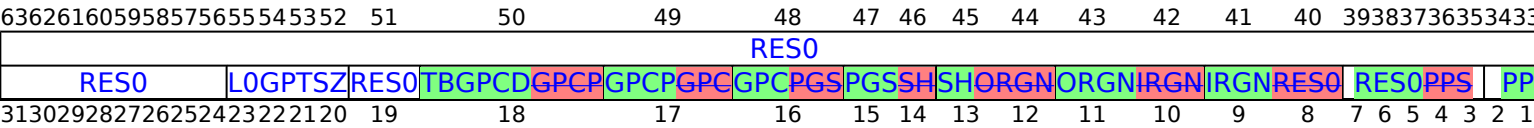
Configuration

This register is present only when FEAT_RME is implemented. Otherwise, direct accesses to GPCCR_EL3 are UNDEFINED.

Attributes

GPCCR_EL3 is a 64-bit register.

Field descriptions



Bits [63:24]

Reserved, RES0.

LOGPTSZ, bits [23:20]

Level 0 GPT entry size.

This field advertises the number of least-significant address bits protected by each entry in the level 0 GPT.

LOGPTSZ	Meaning
0b0000	30-bits. Each entry covers 1GB of address space.
0b0100	34-bits. Each entry covers 16GB of address space.
0b0110	36-bits. Each entry covers 64GB of address space.
0b1001	39-bits. Each entry covers 512GB of address space.

All other values are reserved.

Access to this field is **RO**.

BitBits [19:18]

Reserved, RES0.

TBGPCD, bit [18]**When FEAT_TRBE_EXT is implemented:**

Trace Buffer Granule Protection Check Disabled. Controls whether the Trace Buffer Unit accepts or rejects trace when Granule Protection Checks are disabled.

TBGPCD	Meaning
0b0	The Trace Buffer Unit rejects trace when GPCCR_EL3.GPC is 0.
0b1	The Trace Buffer Unit accepts trace when GPCCR_EL3.GPC is 0.

When the Trace Buffer Unit rejects trace, the trace might remain buffered by the trace unit until the Trace Buffer Unit is able to accept trace. When the Trace Buffer Unit accepts trace, the Trace Buffer Unit writes the trace to memory.

Note

Setting GPCCR_EL3.{TBGPCD, GPC} to {1, 0} means that the Trace Buffer Unit might write to memory without any Granule Protection Checks. The addresses that the Trace Buffer Unit writes to can be programmed by an external agent. The physical address spaces the Trace Buffer Unit can address are restricted by an IMPLEMENTATION DEFINED debug authentication interface.

Setting GPCCR_EL3.{TBGPCD, GPC} to {1, 1} means that GPCCR_EL3.{TBGPCD, GPC} will become {1, 0} on a Warm reset.

This field is ignored by the PE and treated as one when any of the following are true:

- GPCCR_EL3.GPC == 1.
- ExternalRootInvasiveDebugEnabled() == TRUE.

Otherwise:

Reserved, RES0.

GPCP, bit [17]

Granule Protection Check Priority.

This control governs behavior of granule protection checks on fetches of stage 2 Table descriptors.

GPCP	Meaning
0b0	GPC faults are all reported with a priority that is consistent with the GPC being performed on any access to physical address space.
0b1	A GPC fault for the fetch of a Table descriptor for a stage 2 translation table walk might not be generated or reported. All other GPC faults are reported with a priority consistent with the GPC being performed on all accesses to physical address spaces.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GPC, bit [16]

Granule Protection Check Enable.

GPC	Meaning
0b0	Granule protection checks are disabled. Accesses are not prevented by this mechanism.
0b1	All accesses to physical address spaces are subject to granule protection checks, except for fetches of GPT information and accesses governed by the GPCCR_EL3.GPCP control.

If any stage of translation is enabled, this bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

PGS, bits [15:14]

Physical Granule size.

PGS	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

All other values are reserved.

The value of this field is permitted to be cached in a TLB.

Granule sizes not supported for stage 1 and not supported for stage 2, as defined in [ID_AA64MMFR0_EL1](#), are reserved. For example, if [ID_AA64MMFR0_EL1.TGran16](#) == 0b0000 and [ID_AA64MMFR0_EL1.TGran16_2](#) == 0b0001, then the PGS encoding 0b10 is reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH, bits [13:12]

GPT fetch Shareability attribute

SH	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

All other values are reserved.

Fetches of GPT information are made with the Shareability attribute that is configured in this field.

If both ORGN and IRGN are configured with Non-cacheable attributes, it is invalid to configure this field to any value other than 0b10.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN, bits [11:10]

GPT fetch Outer cacheability attribute.

ORGN	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Outer cacheability attributes configured in this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN, bits [9:8]

GPT fetch Inner cacheability attribute.

IRGN	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

Fetches of GPT information are made with the Inner cacheability attributes configured in this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:3]

Reserved, RES0.

PPS, bits [2:0]

Protected Physical Address Size.

The size of the memory region protected by [GPTBR_EL3](#), in terms of the number of least-significant address bits.

PPS	Meaning
0b000	32 bits, 4GB protected address space.
0b001	36 bits, 64GB protected address space.
0b010	40 bits, 1TB protected address space.
0b011	42 bits, 4TB protected address space.
0b100	44 bits, 16TB protected address space.
0b101	48 bits, 256TB protected address space.
0b110	52 bits, 4PB protected address space.

All other values are reserved.

Configuration of this field to a value exceeding the implemented physical address size is invalid.

The value of this field is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing GPCCR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, GPCCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = GPCCR_EL3;
```

MSR GPCCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    GPCCR_EL3 = X[t, 64];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TID5, bit [58]

When FEAT_MTE2 is implemented:

Trap ID group 5. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

AArch64:

- [GMID_EL1](#).

TID5	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 and EL0 accesses to ID group 5 registers are trapped to EL2.

When the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field has an Effective value of 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DCT, bit [57]

When FEAT_MTE2 is implemented:

Default Cacheability Tagging. When HCR_EL2.DC is in effect, controls whether stage 1 translations are treated as Tagged or Untagged.

DCT	Meaning
0b0	Stage 1 translations are treated as Untagged.
0b1	Stage 1 translations are treated as Tagged.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ATA, bit [56]

When FEAT_MTE2 is implemented:

Allocation Tag Access. When HCR_EL2.{E2H,TGE} != {1,1}, controls access to Allocation Tags, System registers for Memory tagging, and prevention of Tag checking, at EL1 and EL0.

ATA	Meaning
0b0	Access to Allocation Tags is prevented at EL1 and EL0. Accesses at EL1 to GCR_EL1 , RGSRR_EL1 , TFSR_EL1 , or TFSR_EL2 , or TFSRE0_EL1 that are not UNDEFINED are trapped to EL2. Accesses at EL1 using MRS or MSR with the register name TFSR_EL2 that are not UNDEFINED are trapped to EL3. Memory accesses at EL1 and EL0 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL1 and EL0. This control does not prevent Tag checking at EL1 and EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TTLBOS, bit [55]

When FEAT_EVT is implemented:

Trap TLB maintenance instructions that operate on the Outer Shareable domain. Traps execution of those TLB maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

[TLBI VMALLE1OS](#), [TLBI VAE1OS](#), [TLBI ASIDE1OS](#), [TLBI VAAE1OS](#), [TLBI VALE1OS](#), [TLBI VAALE1OS](#), [TLBI RVAE1OS](#), [TLBI RVAAE1OS](#), [TLBI RVALE1OS](#), and [TLBI RVAALE1OS](#).

TTLBOS	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TTLBIS, bit [54]

When FEAT_EVT is implemented:

Trap TLB maintenance instructions that operate on the Inner Shareable domain. Traps execution of those TLB maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When EL1 is using AArch64, [TLBI VMALLE1IS](#), [TLBI VAE1IS](#), [TLBI ASIDE1IS](#), [TLBI VAAE1IS](#), [TLBI VALE1IS](#), [TLBI VAALE1IS](#), [TLBI RVAE1IS](#), [TLBI RVAAE1IS](#), [TLBI RVALE1IS](#), and [TLBI RVAALE1IS](#).
- When EL1 is using AArch32, [TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAAIS](#), [TLBIMVALIS](#), and [TLBIMVAALIS](#).

TTLBIS	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnSCXT, bit [53]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Enable Access to the [SCXTNUM_EL1](#) and [SCXTNUM_EL0](#) registers. The defined values are:

EnSCXT	Meaning
0b0	When HCR_EL2.E2H is 0 or HCR_EL2.TGE is 0, and EL2 is enabled in the current Security state, EL1 and EL0 access to SCXTNUM_EL0 and EL1 access to SCXTNUM_EL1 is disabled by this mechanism, causing an exception to EL2, and the values of these registers to be treated as 0. When HCR_EL2.{E2H, TGE} is {1, 1} and EL2 is enabled in the current Security state, EL0 access to SCXTNUM_EL0 is disabled by this mechanism, causing an exception to EL2, and the value of this register to be treated as 0.
0b1	This control does not cause accesses to SCXTNUM_EL0 or SCXTNUM_EL1 to be trapped.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TOCU, bit [52]

When FEAT_EVT is implemented:

Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When [SCTLR_EL1](#).UCI is 1, HCR_EL2.{TGE, E2H} is not {1, 1}, and EL0 is using AArch64, [IC IVAU](#), [DC CVAU](#).
- When EL1 is using AArch64, [IC IVAU](#), [IC IALLU](#), [DC CVAU](#).
- When EL1 is using AArch32, [ICIMVAU](#), [IC IALLU](#), [DCCMVAU](#).

Note

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:

- [IC IALLUIS](#) and [IC IALLU](#) are always UNDEFINED at EL0 using AArch64.
- [ICIMVAU](#), [IC IALLU](#), [IC IALLUIS](#), and [DCCMVAU](#) are always UNDEFINED at EL0 using AArch32.

TOCU	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions are trapped to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AMVOFFEN, bit [51]

When FEAT_AMUv1p1 is implemented:

Activity Monitors Virtual Offsets Enable.

AMVOFFEN	Meaning
0b0	Virtualization of the Activity Monitors is disabled. Indirect reads of the virtual offset registers are zero.
0b1	Virtualization of the Activity Monitors is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TICAB, bit [50]

When FEAT_EVT is implemented:

Trap ICIALLUIS/IC IALLUIS cache maintenance instructions. Traps execution of those cache maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state. This applies to the following instructions:

- When EL1 is using AArch64, [IC IALLUIS](#).
- When EL1 is using AArch32, [ICIALLUIS](#).

TICAB	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 execution of the specified instructions is trapped to EL2.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TID4, bit [49]**When FEAT_EVT is implemented:**

Trap ID group 4. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

AArch64:

- EL1 reads of [CCSIDR_EL1](#), [CCSIDR2_EL1](#), [CLIDR_EL1](#), and [CSSELR_EL1](#).
- EL1 writes to [CSSELR_EL1](#).

AArch32:

- EL1 reads of [CCSIDR](#), [CCSIDR2](#), [CLIDR](#), and [CSSELR](#).
- EL1 writes to [CSSELR](#).

TID4	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 and EL0 accesses to ID group 4 registers are trapped to EL2.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

GPF, bit [48]**When FEAT_RME is implemented:**

Controls the reporting of Granule protection faults at EL0 and EL1.

GPF	Meaning
0b0	This control does not cause exceptions to be routed from EL0 and EL1 to EL2.
0b1	Instruction Abort exceptions and Data Abort exceptions due to GPFs from EL0 and EL1 are routed to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FIEN, bit [47]
When FEAT_RASv1p1 is implemented:

Fault Injection Enable. Unless this bit is set to 1, accesses to the [ERXPGCDN_EL1](#), [ERXPGCTL_EL1](#), and [ERXPFGF_EL1](#) registers from EL1 generate a Trap exception to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18.

FIEN	Meaning
0b0	Accesses to the specified registers from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.
0b1	This control does not cause any instructions to be trapped.

If EL2 is disabled in the current Security state, the Effective value of HCR_EL2.FIEN is 0b1.

If [ERRIDR_EL1](#).NUM is zero, meaning no error records are implemented, or no error record accessible using System registers is owned by a node that implements the RAS Common Fault Injection Model Extension, then this bit might be RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FWB, bit [46]
When FEAT_S2FWB is implemented:

Forced Write-Back. Defines the combined cacheability attributes in a 2 stage translation regime.

Note
When FEAT_MTE2 is implemented, if the stage 1 page or block descriptor specifies the Tagged attribute, the final memory type is Tagged only if the final cacheable memory type is Inner and Outer Write-back cacheable and the final allocation hints are Read-Allocate, Write-Allocate.

FWB	Meaning
0b0	<p>When this bit is 0, then:</p> <ul style="list-style-type: none"> The combination of stage 1 and stage 2 translations on memory type and cacheability attributes are as described in the Armv8.0 architecture. For more information, see 'Combining the stage 1 and stage 2 memory attributes, type EL1&0 attributes translation regime'. The encoding of the stage 2 memory type and cacheability attributes in bits[5:2] of the stage 2 page or block descriptors are as described in the Armv8.0 architecture.
0b1	<p>When this bit is 1, then:</p> <ul style="list-style-type: none"> Bit[5] of stage 2 page or block descriptor is RES0. When bit[4] of stage 2 page or block descriptor is 1 and when: <ul style="list-style-type: none"> Bits[3:2] of stage 2 page or block descriptor are 0b11, the resultant memory type and inner or outer cacheability attribute is the same as the stage 1 memory type and inner or outer cacheability attribute. Bits[3:2] of stage 2 page or block descriptor are 0b10, the resultant memory type and attribute is Normal Write-Back. Bits[3:2] of stage 2 page or block descriptor are 0b0x, the resultant memory type will be Normal Non-cacheable except where the stage 1 memory type was Device-<attr> the resultant memory type will be Device-<attr> When bit[4] of stage 2 page or block descriptor is 0 the memory type is Device, and when: <ul style="list-style-type: none"> Bits[3:2] of stage 2 page or block descriptor are 0b00, the stage 2 memory type is Device-nGnRnE. Bits[3:2] of stage 2 page or block descriptor are 0b01, the stage 2 memory type is Device-nGnRE. Bits[3:2] of stage 2 page or block descriptor are 0b10, the stage 2 memory type is Device-nGRE. Bits[3:2] of stage 2 page or block descriptor are 0b11, the stage 2 memory type is Device-GRE. If the stage 1 translation specifies a cacheable memory type, then the stage 1 cache allocation hint is applied to the final cache allocation hint where the final memory type is cacheable. If the stage 1 translation does not specify a cacheable memory type, then if the final memory type is cacheable, it is treated as read allocate, write allocate. For the more stage information, 1 see and stage 2 memory types are combined in the manner described in 'Stage Combining 2 the memory stage type 1 and Cacheability stage 2 attributes, when EL1&0 FEAT_S2FWB translation is enabled regime'.

In Secure state, this bit applies to both the Secure stage 2 translation and the Non-secure stage 2 translation.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NV2, bit [45]**When FEAT_NV2 is implemented:**

Nested Virtualization. Changes the behaviors of HCR_EL2.{NV1, NV} to provide a mechanism for hardware to transform reads and writes from System registers into reads and writes from memory.

NV2	Meaning
0b0	This bit has no effect on the behavior of HCR_EL2.{NV1, NV}. The behavior of HCR_EL2.{NV1, NV} is as defined for FEAT_NV.
0b1	Redefines behavior of HCR_EL2{NV1, NV} to enable: <ul style="list-style-type: none"> Transformation of read/writes to registers into read/writes to memory. Redirection of EL2 registers to EL1 registers. Any exception taken from EL1 and taken to EL1 causes SPSR_EL1.M[3:2] to be set to 0b10 and not 0b01.

When HCR_EL2.NV is 0, the Effective value of this field is 0 and this field is treated as 0 for all purposes other than direct reads and writes of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AT, bit [44]**When FEAT_NV is implemented:**

Address Translation. EL1 execution of the following address translation instructions is trapped to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18:

- [AT S1E0R](#), [AT S1E0W](#), [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#).

AT	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 execution of the specified instructions is trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NV1, bit [43]**When FEAT_NV2 is implemented:**

Nested Virtualization.

NV1	Meaning
0b0	If HCR_EL2.{NV2, NV} are both 1, accesses executed from EL1 to implemented EL12, EL02, or EL2 registers are transformed to loads and stores. If HCR_EL2.NV2 is 0 or HCR_EL2.{NV2, NV} == {1, 0}, this control does not cause any instructions to be trapped.
0b1	If HCR_EL2.NV2 is 1, accesses executed from EL1 to implemented EL2 registers are transformed to loads and stores. If HCR_EL2.NV2 is 0, EL1 accesses to VBAR_EL1 , ELR_EL1 , SPSR_EL1 , and, when FEAT_CSV2_2 or FEAT_CSV2_1p2 is implemented, SCXTNUM_EL1 , are trapped to EL2, when EL2 is enabled in the current Security state, and are reported using EC syndrome value 0x18.

If HCR_EL2.NV2 is 1, the value of HCR_EL2.NV1 defines which EL1 register accesses are transformed to loads and stores. These transformed accesses have priority over the trapping of registers.

The trapping of EL1 registers caused by other control bits has priority over the transformation of these accesses.

If a register is specified that is not implemented by an implementation, then access to that register are UNDEFINED.

For the list of registers affected, see 'Enhanced support for nested virtualization'.

If HCR_EL2.{NV1, NV} is {0, 1}, any exception taken from EL1, and taken to EL1, causes the [SPSR_EL1](#).M[3:2] to be set to 0b10, and not 0b01.

If HCR_EL2.{NV1, NV} is {1, 1}, then:

- The EL1 translation table Block and Page descriptors:
 - Bit[54] holds the PXN instead of the UXN.
 - Bit[53] is RES0.
 - Bit[6] is treated as 0 regardless of the actual value.
- If Hierarchical Permissions are enabled, the EL1 translation table Table descriptors are as follows:
 - Bit[61] is treated as 0 regardless of the actual value.
 - Bit[60] holds the PXNTable instead of the UXNTable.
 - Bit[59] is RES0.
- When executing at EL1, the PSTATE.PAN bit is treated as zero for all purposes except reading the value of the bit.
- When executing at EL1, the LDTR* instructions are treated as the equivalent LDR* instructions, and the STTR* instructions are treated as the equivalent STR* instructions.

If HCR_EL2.{NV1, NV} are {1, 0}, then the behavior is a CONSTRAINED UNPREDICTABLE choice of:

- Behaving as if HCR_EL2.NV is 1 and HCR_EL2.NV1 is 1 for all purposes other than reading back the value of the HCR_EL2.NV bit.
- Behaving as if HCR_EL2.NV is 0 and HCR_EL2.NV1 is 0 for all purposes other than reading back the value of the HCR_EL2.NV1 bit.
- Behaving with regard to the HCR_EL2.NV and HCR_EL2.NV1 bits behavior as defined in the rest of this description.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_NV is implemented:

Nested Virtualization. EL1 accesses to certain registers are trapped to EL2, when EL2 is enabled in the current Security state.

NV1	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to VBAR_EL1 , ELR_EL1 , SPSR_EL1 , and, when FEAT_CSV2_2 or FEAT_CSV2_1p2 is implemented, SCXTNUM_EL1 , are trapped to EL2, when EL2 is enabled in the current Security state, and are reported using EC syndrome value 0x18.

If HCR_EL2.NV is 1 and HCR_EL2.NV1 is 0, then the following effects also apply:

- Any exception taken from EL1, and taken to EL1, causes the [SPSR_EL1](#).M[3:2] to be set to 0b10, and not 0b01.

If HCR_EL2.NV and HCR_EL2.NV1 are both set to 1, then the following effects also apply:

- The EL1 translation table Block and Page descriptors:
 - Bit[54] holds the PXN instead of the UXN.
 - Bit[53] is RES0.
 - Bit[6] is treated as 0 regardless of the actual value.
- If Hierarchical Permissions are enabled, the EL1 translation table Table descriptors are as follows:
 - Bit[61] is treated as 0 regardless of the actual value.
 - Bit[60] holds the PXNTable instead of the UXNTable.
 - Bit[59] is RES0.
- When executing at EL1, the PSTATE.PAN bit is treated as zero for all purposes except reading the value of the bit.
- When executing at EL1, the LDTR* instructions are treated as the equivalent LDR* instructions, and the STTR* instructions are treated as the equivalent STR* instructions.

If HCR_EL2.NV is 0 and HCR_EL2.NV1 is 1, then the behavior is a CONSTRAINED UNPREDICTABLE choice of:

- Behaving as if HCR_EL2.NV is 1 and HCR_EL2.NV1 is 1 for all purposes other than reading back the value of the HCR_EL2.NV bit.
- Behaving as if HCR_EL2.NV is 0 and HCR_EL2.NV1 is 0 for all purposes other than reading back the value of the HCR_EL2.NV1 bit.
- Behaving with regard to the HCR_EL2.NV and HCR_EL2.NV1 bits behavior as defined in the rest of this description.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NV, bit [42]

When FEAT_NV2 is implemented:

Nested Virtualization.

When HCR_EL2.NV2 is 1, redefines register accesses so that:

- Instructions accessing the Special purpose registers [SPSR_EL2](#) and [ELR_EL2](#) instead access [SPSR_EL1](#) and [ELR_EL1](#) respectively.
- Instructions accessing the System registers [ESR_EL2](#) and [FAR_EL2](#) instead access [ESR_EL1](#) and [FAR_EL1](#).

When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, traps functionality that is permitted at EL2 and would be UNDEFINED at EL1 if this field was 0, when EL2 is enabled in the current Security state. This applies to the following operations:

- EL1 accesses to Special-purpose registers that are not UNDEFINED at EL2.
- EL1 accesses to System registers that are not UNDEFINED at EL2.
- Execution of EL1 or EL2 translation regime address translation and TLB maintenance instructions for EL2 and above.

NV	Meaning
0b0	When this bit is set to 0, then the PE behaves as if HCR_EL2.NV2 is 0 for all purposes other than reading this register. This control does not cause any instructions to be trapped. When HCR_EL2.NV2 is 1, no FEAT_NV2 functionality is implemented.
0b1	When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, EL1 accesses to the specified registers or the execution of the specified instructions are trapped to EL2, when EL2 is enabled in the current Security state. EL1 read accesses to the CurrentEL register return a value of 0x2. When HCR_EL2.NV2 is 1, this control redefines EL1 register accesses so that instructions accessing SPSR_EL2 , ELR_EL2 , ESR_EL2 , and FAR_EL2 instead access SPSR_EL1 , ELR_EL1 , ESR_EL1 , and FAR_EL1 respectively.

When HCR_EL2.NV2 is 0, or if FEAT_NV2 is not implemented, then:

- The System or Special-purpose registers for which accesses are trapped and reported using EC syndrome value 0x18 are as follows:
 - Registers accessed using MRS or MSR with a name ending in _EL2, except [SP_EL2](#).
 - Registers accessed using MRS or MSR with a name ending in _EL12.
 - Registers accessed using MRS or MSR with a name ending in _EL02.
 - Special-purpose registers [SPSR_irq](#), [SPSR_abt](#), [SPSR_und](#) and [SPSR_fiq](#), accessed using MRS or MSR.
 - Special-purpose register [SP_EL1](#) accessed using the dedicated MRS or MSR instruction.
- The instructions for which the execution is trapped and reported using EC syndrome value 0x18 are as follows:
 - EL2 translation regime Address Translation instructions and TLB maintenance instructions.
 - EL1 translation regime Address Translation instructions and TLB maintenance instructions that are accessible only from EL2 and EL3.
- The instructions for which the execution is trapped as follows:
 - SMC in an implementation that does not include EL3 and when HCR_EL2.TSC is 1. HCR_EL2.TSC bit is not RES0 in this case. This is reported using EC syndrome value 0x17.
 - The ERET, ERETAA, and ERETAB instructions, reported using EC syndrome value 0x1A.

Note

The priority of this trap is higher than the priority of the HCR_EL2.API trap. If both of these bits are set so that EL1 execution of an ERETAA or ERETAB instruction is trapped to EL2, then the syndrome reported is 0x1A.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_NV is implemented:

Nested Virtualization. Traps functionality that is permitted at EL2 and would be UNDEFINED at EL1 if this field was 0, when EL2 is enabled in the current Security state. This applies to the following operations:

- EL1 accesses to Special-purpose registers that are not UNDEFINED at EL2.
- EL1 accesses to System registers that are not UNDEFINED at EL2.
- Execution of EL1 or EL2 translation regime address translation and TLB maintenance instructions for EL2 and above.

NV	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to the specified registers or the execution of the specified instructions are trapped to EL2, when EL2 is enabled in the current Security state. EL1 read accesses to the CurrentEL register return a value of 0x2.

The System or Special-purpose registers for which accesses are trapped and reported using EC syndrome value 0x18 are as follows:

- Registers accessed using MRS or MSR with a name ending in _EL2, except [SP_EL2](#).

- Registers accessed using MRS or MSR with a name ending in `_EL12`.
- Registers accessed using MRS or MSR with a name ending in `_EL02`.
- Special-purpose registers [SPSR_irq](#), [SPSR_abt](#), [SPSR_und](#) and [SPSR_fiq](#), accessed using MRS or MSR.
- Special-purpose register [SP_EL1](#) accessed using the dedicated MRS or MSR instruction.

The instructions for which the execution is trapped and reported using EC syndrome value `0x18` are as follows:

- EL2 translation regime Address Translation instructions and TLB maintenance instructions.
- EL1 translation regime Address Translation instructions and TLB maintenance instructions that are accessible only from EL2 and EL3.

The execution of the ERET, ERETAA, and ERETAB instructions are trapped and reported using EC syndrome value `0x1A`.

Note

The priority of this trap is higher than the priority of the HCR_EL2.API trap. If both of these bits are set so that EL1 execution of an ERETAA or ERETAB instruction is trapped to EL2, then the syndrome reported is `0x1A`.

The execution of the SMC instructions in an implementation that does not include EL3 and when HCR_EL2.TSC is 1 are trapped and reported using EC syndrome value `0x17`. HCR_EL2.TSC bit is not RES0 in this case.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

API, bit [41]

When FEAT_PAuth is implemented:

Controls the use of instructions related to Pointer Authentication:

- In EL0, when HCR_EL2.TGE==0 or HCR_EL2.E2H==0, and the associated [SCTLR_EL1.En<N><M>==1](#).
- In EL1, the associated [SCTLR_EL1.En<N><M>==1](#).

Traps are reported using EC syndrome value `0x09`. The Pointer Authentication instructions trapped are:

- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB.
- PACGA, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZB.
- RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ.
- ERETAA, ERETAB, LDRAA, and LDRAB.

API	Meaning
0b0	<p>The instructions related to Pointer Authentication are trapped to EL2, when EL2 is enabled in the current Security state and the instructions are enabled for the EL1&0 translation regime, from:</p> <ul style="list-style-type: none"> • EL0 when HCR_EL2.TGE==0 or HCR_EL2.E2H==0. • EL1. <p>If HCR_EL2.NV is 1, the HCR_EL2.NV trap takes precedence over the HCR_EL2.API trap for the ERETAA and ERETAB instructions.</p> <p>If EL2 is implemented and enabled in the current Security state and HFGITR_EL2.ERET == 1, execution at EL1 using AArch64 of ERETAA or ERETAB instructions is reported with EC syndrome value <code>0x1A</code> with its associated ISS field, as the fine-grained trap has higher priority than the HCR_EL2.API == 0.</p>
0b1	This control does not cause any instructions to be trapped.

If FEAT_PAuth is implemented but EL2 is not implemented or disabled in the current Security state, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

APK, bit [40]

When FEAT_PAuth is implemented:

Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers from EL1 to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18:

- [APIAKeyLo_EL1](#), [APIAKeyHi_EL1](#), [APIBKeyLo_EL1](#), [APIBKeyHi_EL1](#), [APDAKeyLo_EL1](#), [APDAKeyHi_EL1](#), [APDBKeyLo_EL1](#), [APDBKeyHi_EL1](#), [APGAKeyLo_EL1](#), and [APGAKeyHi_EL1](#).

APK	Meaning
0b0	Access to the registers holding "key" values for pointer authentication from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.
0b1	This control does not cause any instructions to be trapped.

Note

If FEAT_PAuth is implemented but EL2 is not implemented or is disabled in the current Security state, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TME, bit [39]

When FEAT_TME is implemented:

Enables access to the TSTART, TCOMMIT, TTEST, and TCANCEL instructions at EL0 and EL1.

TME	Meaning
0b0	EL0 and EL1 accesses to TSTART, TCOMMIT, TTEST, and TCANCEL instructions are UNDEFINED.
0b1	This control does not cause any instruction to be UNDEFINED.

If EL2 is not implemented or is disabled in the current Security state, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MIOCNCE, bit [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the EL1&0 translation regimes.

MIOCNCE	Meaning
0b0	For the EL1&0 translation regimes, for permitted accesses to a memory location that use a common definition of the Shareability and Cacheability of the location, there must be no loss of coherency if the Inner Cacheability attribute for those accesses differs from the Outer Cacheability attribute.
0b1	For the EL1&0 translation regimes, for permitted accesses to a memory location that use a common definition of the Shareability and Cacheability of the location, there might be a loss of coherency if the Inner Cacheability attribute for those accesses differs from the Outer Cacheability attribute.

For more information, see 'Mismatched memory attributes'.

This field can be implemented as RAZ/WI.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TEA, bit [37]**When FEAT_RAS is implemented:**

Route synchronous External abort exceptions to EL2.

TEA	Meaning
0b0	This control does not cause exceptions to be routed from EL0 and EL1 to EL2.
0b1	Route synchronous External abort exceptions from EL0 and EL1 to EL2, when EL2 is enabled in the current Security state, if not routed to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TERR, bit [36]**When FEAT_RAS is implemented:**

Trap Error record accesses. Trap accesses to the RAS error registers from EL1 to EL2 as follows:

- If EL1 is using AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18:
 - [ERRIDR_EL1](#), [ERRSELR_EL1](#), [ERXADDR_EL1](#), [ERXCTLR_EL1](#), [ERXFR_EL1](#), [ERXMISC0_EL1](#), [ERXMISC1_EL1](#), and [ERXSTATUS_EL1](#).
 - When FEAT_RASv1p1 is implemented, [ERXMISC2_EL1](#), and [ERXMISC3_EL1](#).
- If EL1 is using AArch32 state, MCR or MRC accesses are trapped to EL2, reported using EC syndrome value 0x03, MCRR or MRRC accesses are trapped to EL2, reported using EC syndrome value 0x04:
 - [ERRIDR](#), [ERRSELR](#), [ERXADDR](#), [ERXADDR2](#), [ERXCTLR](#), [ERXCTLR2](#), [ERXFR](#), [ERXFR2](#), [ERXMISC0](#), [ERXMISC1](#), [ERXMISC2](#), [ERXMISC3](#), and [ERXSTATUS](#).
 - When FEAT_RASv1p1 is implemented, [ERXMISC4](#), [ERXMISC5](#), [ERXMISC6](#), and [ERXMISC7](#).

TERR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Accesses to the specified registers from EL1 generate a Trap exception to EL2, when EL2 is enabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TLOR, bit [35]

When FEAT_LOR is implemented:

Trap LOR registers. Traps Non-secure EL1 accesses to [LORSA_EL1](#), [LOREA_EL1](#), [LORN_EL1](#), [LORC_EL1](#), and [LORID_EL1](#) registers to EL2.

TLOR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 accesses to the LOR registers are trapped to EL2.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E2H, bit [34]

When FEAT_VHE is implemented:

EL2 Host. Enables a configuration where a Host Operating System is running in EL2, and the Host Operating System's applications are running in EL0.

E2H	Meaning
0b0	The facilities to support a Host Operating System at EL2 are disabled.
0b1	The facilities to support a Host Operating System at EL2 are enabled.

For information on the behavior of this bit see 'Behavior of HCR_EL2.E2H'.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ID, bit [33]

Stage 2 Instruction access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

ID	Meaning
0b0	This control has no effect on stage 2 of the EL1&0 translation regime.
0b1	Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CD, bit [32]

Stage 2 Data access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

CD	Meaning
0b0	This control has no effect on stage 2 of the EL1&0 translation regime for data accesses and translation table walks.
0b1	Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

RW, bit [31]**When EL1 is capable of using AArch32:**

Execution state control for lower Exception levels:

RW	Meaning
0b0	Lower levels are all AArch32.
0b1	The Execution state for EL1 is AArch64. The Execution state for EL0 is determined by the current value of PSTATE.nRW when executing at EL0.

In an implementation that includes EL3, when EL2 is not enabled in Secure state, the PE behaves as if this bit has the same value as the [SCR_EL3.RW](#) bit for all purposes other than a direct read or write access of HCR_EL2.

The RW bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 1 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAO/WI.

TRVM, bit [30]

Trap Reads of Virtual Memory controls. Traps EL1 reads of the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, EL1 accesses to the following registers are trapped to EL2 and reported using EC syndrome value 0x18 for MRS and 0x14 for MRRS:
 - SCTL_EL1, TTBR0_EL1, TTBR1_EL1, TCR_EL1, ESR_EL1, FAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR_EL1, AMAIR_EL1, CONTEXTIDR_EL1.
 - SCTL_EL1, TTBR0_EL1, TTBR1_EL1, TCR_EL1, ESR_EL1, FAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR_EL1, AMAIR_EL1, CONTEXTIDR_EL1.
 - If FEAT_AIE is implemented, MAIR2_EL1, AMAIR2_EL1.
 - If FEAT_S1PIE is implemented, PIRE0_EL1, PIR_EL1.
 - If FEAT_S1POE is implemented, POR_EL0, POR_EL1.
 - If FEAT_S2POE is implemented, S2POR_EL1.
 - If FEAT_TCR2 is implemented, TCR2_EL1.
 - If FEAT_SCTLR2 is implemented, SCTLR2_EL1.
- If HCR_EL2.{E2H, EL1 TGE} is not {1, 1}, and EL0 is using AArch64 state, EL0 accesses using MRC to the following registers are trapped to EL2 and reported using EC syndrome value 0x18 for accesses using MRC, accesses using MRRC are trapped to EL2 and reported using EC syndrome value 0x03 for accesses using MRRC are trapped to EL2 and reported using EC syndrome value 0x04:
 - SCTL_EL1, TTBR0_EL1, TTBR1_EL1, TTBCR_EL1, TTBCR2_EL1, DACR_EL1, DFSR_EL1, IFSR_EL1, DFAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR0_EL1, MAIR1_EL1, AMAIR0_EL1, AMAIR1_EL1, CONTEXTIDR_EL1, POR_EL0, TTBR0_EL1.
- If EL1 is using AArch32 state, EL1 accesses using MRC to the following registers are trapped to EL2 and reported using EC syndrome value 0x03, accesses using MRRC are trapped to EL2 and reported using EC syndrome value 0x04:
 - SCTL_EL1, TTBR0_EL1, TTBR1_EL1, TTBCR_EL1, TTBCR2_EL1, DACR_EL1, DFSR_EL1, IFSR_EL1, DFAR_EL1, AFSR0_EL1, AFSR1_EL1, MAIR0_EL1, MAIR1_EL1, AMAIR0_EL1, AMAIR1_EL1, CONTEXTIDR_EL1, POR_EL0, TTBR0_EL1.

TRVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Read EL1 read accesses to the specified Virtual Memory control registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.{E2H, HCR_EL2.TGE} is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

Note

EL2 provides a second stage of address translation, that a hypervisor can use to remap the address map defined by a Guest OS. In addition, a hypervisor can trap attempts by a Guest OS to write to the registers that control the memory system. A hypervisor might use this trap as part of its virtualization of memory management.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HCD, bit [29]**When EL3 is not implemented:**

HVC instruction disable. Disables EL1 execution of HVC instructions, from both Execution states, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x00.

HCD	Meaning
0b0	HVC instruction execution is enabled at EL2 and EL1.
0b1	HVC instructions are UNDEFINED at EL2 and EL1. Any resulting exception is taken to the Exception level at which the HVC instruction is executed.

Note

HVC instructions are always UNDEFINED at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TDZ, bit [28]

Trap [DC ZVA](#) instructions. Traps EL0 and EL1 execution of [DC ZVA](#) instructions to EL2, when EL2 is enabled in the current Security state, from AArch64 state only, reported using EC syndrome value 0x18.

If FEAT_MTE is implemented, this trap also applies to [DC GVA](#) and [DC GZVA](#).

TDZ	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	In AArch64 state, any attempt to execute an instruction this trap applies to at EL1, or at EL0 when the instruction is not UNDEFINED at EL0, is trapped to EL2 when EL2 is enabled in the current Security state. Reading the DCZID_EL0 returns a value that indicates that the instructions this trap applies to are not supported.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TGE, bit [27]

Trap General Exceptions, from EL0.

TGE	Meaning
0b0	This control has no effect on execution at EL0.
0b1	<p>When EL2 is not enabled in the current Security state, this control has no effect on execution at EL0.</p> <p>When EL2 is enabled in the current Security state, in all cases:</p> <ul style="list-style-type: none"> • All exceptions that would be routed to EL1 are routed to EL2. • If EL1 is using AArch64, the SCTLR_EL1.M field is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR_EL1. • If EL1 is using AArch32, the SCTLR.M field is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR. • All virtual interrupts are disabled. • Any IMPLEMENTATION DEFINED mechanisms for signaling virtual interrupts are disabled. • An exception return to EL1 is treated as an illegal exception return. • The MDCR_EL2.{TDRA, TDOSA, TDA, TDE} fields are treated as being 1 for all purposes other than returning the result of a direct read of MDCR_EL2. <p>In addition, when EL2 is enabled in the current Security state, if:</p> <ul style="list-style-type: none"> • HCR_EL2.E2H is 0, the Effective values of the HCR_EL2.{FMO, IMO, AMO} fields are 1. • HCR_EL2.E2H is 1, the Effective values of the HCR_EL2.{FMO, IMO, AMO} fields are 0. <p>For further information on the behavior of this bit when E2H is 1, see 'Behavior of HCR_EL2.E2H'.</p>

HCR_EL2.TGE must not be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TVM, bit [26]

Trap Virtual Memory controls. Traps [EL1](#) writes to the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, the following registers are trapped to EL2 and reported using EC syndrome value 0x18 for [MSR](#) and 0x14 for [MSRR](#):
 - [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).
 - [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).
 - If FEAT_AIE is implemented, [MAIR2_EL1](#), [AMAIR2_EL1](#).
 - If FEAT_S1PIE is implemented, [PIRE0_EL1](#), [PIR_EL1](#).
 - If FEAT_S1POE is implemented, [POR_EL0](#), [POR_EL1](#).
 - If FEAT_S2POE is implemented, [S2POR_EL1](#).
 - If FEAT_TCR2 is implemented, [TCR2_EL1](#).
 - If FEAT_SCTLR2 is implemented, [SCTLR2_EL1](#).
- If HCR_EL2.{E2H, [EL1](#) TGE} is not {1, 1}, and EL0 is using [AArch64](#) [AArch32](#) state, [EL0](#) accesses [accesses using MCR](#) to the following registers are trapped to EL2 and reported using EC syndrome value 0x18 [0x03](#) for [accesses using MCRR](#) are trapped to EL2 and reported using EC syndrome value [MSR0X04](#):
 - [SCTLR](#) If FEAT_S1POE is implemented, [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIR0](#), [AMAIR1](#), [CONTEXTIDR](#), [POR_EL0](#), [TTBR0](#).

- If EL1 is using AArch32 state, EL1 accesses using MCR to the following registers are trapped to EL2 and reported using EC syndrome value 0x03, accesses using MCRR are trapped to EL2 and reported using EC syndrome value 0x04:
 - [SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIRO](#), [AMAIR1](#), [CONTEXTIDR](#).

TVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	WriteEL1 write accesses to the specified VirtualEL1 Memoryvirtual memory control registers are trapped to EL2, when EL2 is enabled in the current Security state.

When [HCR_EL2.{E2H,HCR_EL2.TGE TGE}](#) is {1, 1}, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TTLB, bit [25]

Trap TLB maintenance instructions. Traps EL1 execution of TLB maintenance instructions to EL2, when EL2 is enabled in the current Security state, as follows:

- When EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
 - [TLBI VMALLE1](#), [TLBI VAE1](#), [TLBI ASIDE1](#), [TLBI VAAE1](#), [TLBI VALE1](#), [TLBI VAALE1](#).
 - [TLBI VMALLE1IS](#), [TLBI VAE1IS](#), [TLBI ASIDE1IS](#), [TLBI VAAE1IS](#), [TLBI VALE1IS](#), [TLBI VAALE1IS](#).
 - If FEAT_TLBIOS is implemented, this trap applies to [TLBI VMALLE1OS](#), [TLBI VAE1OS](#), [TLBI ASIDE1OS](#), [TLBI VAAE1OS](#), [TLBI VALE1OS](#), [TLBI VAALE1OS](#).
 - If FEAT_TLBIRANGE is implemented, this trap applies to [TLBI RVAE1](#), [TLBI RVAAE1](#), [TLBI RVALE1](#), [TLBI RVALE1](#), [TLBI RVAE1IS](#), [TLBI RVAAE1IS](#), [TLBI RVALE1IS](#), [TLBI RVALE1IS](#).
 - If FEAT_TLBIOS and FEAT_TLBIRANGE are implemented, this trap applies to [TLBI RVAE1OS](#), [TLBI RVAAE1OS](#), [TLBI RVALE1OS](#), [TLBI RVALE1OS](#).
- When EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x03:
 - [TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAASIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#).
 - [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [TLBIMVAA](#), [TLBIMVAL](#), [TLBIMVAAL](#).
 - [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#).
 - [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#).

TTLB	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 execution of the specified TLB maintenance instructions are trapped to EL2, when EL2 is enabled in the current Security state.

When [HCR_EL2.TGE](#) is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

Note

The TLB maintenance instructions are UNDEFINED at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TPU, bit [24]

Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of [SCTLR_EL1](#).UCI is not 0, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
 - [IC IVAU](#), [DC CVAU](#). If the value of [SCTLR_EL1](#).UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
 - [IC IVAU](#), [IC IALLU](#), [IC IALLUIS](#), [DC CVAU](#).
- If EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18:
 - [ICIMVAU](#), [ICIALLU](#), [ICIALLUIS](#), [DCCMVAU](#).

Note

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:

- [IC IALLUIS](#) and [IC IALLU](#) are always UNDEFINED at EL0 using AArch64.
- [ICIMVAU](#), [ICIALLU](#), [ICIALLUIS](#), and [DCCMVAU](#) are always UNDEFINED at EL0 using AArch32.

TPU	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate to the Point of Unification instruction can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit[23]**When FEAT_DPB is implemented:****TPCP, bit [0] of bit [23]**

Trap data or unified cache maintenance instructions that operate to the Point of Coherency or Persistence. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of [SCTLR_EL1](#).UCI is not 0, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
 - [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#). If the value of [SCTLR_EL1](#).UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18:
 - [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#).
- If EL1 is using AArch32 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x03:
 - [DCIMVAC](#), [DCCIMVAC](#), [DCCMVAC](#).

If FEAT_DPB2 is implemented, this trap also applies to [DC CVADP](#).

If FEAT_MTE is implemented, this trap also applies to [DC CIGVAC](#), [DC CIGDVAC](#), [DC IGVAC](#), [DC IGDVAC](#), [DC CGVAC](#), [DC CGDVAC](#), [DC CGVAP](#) and [DC CGDVAP](#).

If FEAT_DPB2 and FEAT_MTE are implemented, this trap also applies to [DC CGVADP](#) and [DC CGDVADP](#).

Note

- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
 - AArch64 instructions which invalidate by VA to the Point of Coherency are always UNDEFINED at EL0 using AArch64.
 - [DCIMVAC](#), [DCCIMVAC](#), and [DCCMVAC](#) are always UNDEFINED at EL0 using AArch32.
- In Armv8.0 and Armv8.1, this field is named TPC. From Armv8.2, it is named TPCP.

TPCP	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, invalidate, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

If HCR_EL2.{E2H, TGE} is set to {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

TPC, bit [0] of bit [23]

Trap data or unified cache maintenance instructions that operate to the Point of Coherency. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:

- If EL0 is using AArch64 state and the value of [SCTLR_EL1](#).UCI is not 0, accesses to the following registers are trapped and reported using EC syndrome value 0x18:
 - [DC CIVAC](#), [DC CVAC](#). However, if the value of [SCTLR_EL1](#).UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2.
- If EL1 is using AArch64 state, accesses to [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#) are trapped and reported using EC syndrome value 0x18.
- When EL1 is using AArch32, accesses to [DCIMVAC](#), [DCCIMVAC](#), and [DCCMVAC](#) are trapped and reported using EC syndrome value 0x03.

Note

- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
 - AArch64 instructions which invalidate by VA to the Point of Coherency are always UNDEFINED at EL0 using AArch64.
 - [DCIMVAC](#), [DCCIMVAC](#), and [DCCMVAC](#) are always UNDEFINED at EL0 using AArch32.
- In Armv8.0 and Armv8.1, this field is named TPC. From Armv8.2, it is named TPCP.

TPC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, invalidate, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TSW, bit [22]

Trap data or unified cache maintenance instructions that operate by Set/Way. Traps execution of those cache maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state as follows:

- If EL1 is using AArch64 state, accesses to [DC ISW](#), [DC CSW](#), [DC CISW](#) are trapped to EL2, reported using EC syndrome value 0x18.
- If EL1 is using AArch32 state, accesses to [DCISW](#), [DCCSW](#), [DCCISW](#) are trapped to EL2, reported using EC syndrome value 0x03.

If FEAT_MTE2 is implemented, this trap also applies to [DC IGSW](#), [DC IGDSW](#), [DC CGSW](#), [DC CGDW](#), [DC CIGSW](#), and [DC CIGDSW](#).

Note

An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TSW	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TACR, bit [21]

Trap Auxiliary Control Registers. Traps EL1 accesses to the Auxiliary Control Registers to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64 state, accesses to [ACTLR_EL1](#) to EL2, are trapped to EL2 and reported using EC syndrome value 0x18.
- If EL1 is using AArch32 state, accesses to [ACTLR](#) and, if implemented, [ACTLR2](#) are trapped to EL2 and reported using EC syndrome value 0x03.

TACR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to the specified registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

Note

[ACTLR_EL1](#) is not accessible at EL0.

[ACTLR](#) and [ACTLR2](#) are not accessible at EL0.

The Auxiliary Control Registers are IMPLEMENTATION DEFINED registers that might implement global control bits for the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TIDCP, bit [20]

Trap IMPLEMENTATION DEFINED functionality. Traps EL1 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL2, when EL2 is enabled in the current Security state as follows:

- In AArch64 state, access to any of the encodings in the following reserved encoding spaces are trapped and reported using EC syndrome 0x18:
 - IMPLEMENTATION DEFINED System instructions, which are accessed using SYS and SYSL, with CRn == {11, 15}.
 - IMPLEMENTATION DEFINED System registers, which are accessed using MRS and MSR with the [S3_<op1>_<Cn>_<Cm>_<op2>](#) register name.
- In AArch32 state, MCR and MRC access to instructions with the following encodings are trapped and reported using EC syndrome 0x03:
 - All coproc==p15, CRn==c9, opc1 == {0-7}, CRm == {c0-c2, c5-c8}, opc2 == {0-7}.
 - All coproc==p15, CRn==c10, opc1 == {0-7}, CRm == {c0, c1, c4, c8}, opc2 == {0-7}.
 - All coproc==p15, CRn==c11, opc1=={0-7}, CRm == {c0-c8, c15}, opc2 == {0-7}.

When this functionality is accessed from EL0:

- If FEAT_TIDCP1 is implemented and the Effective value of [SCTLR_EL1.TIDCP](#) is 1, any accesses from EL0 are trapped to EL1.
- Otherwise, if FEAT_TIDCP1 is implemented and the Effective value of [SCTLR_EL2.TIDCP](#) is 1, any accesses from EL0 are trapped to EL2.
- Otherwise:
 - If HCR_EL2.TIDCP is 1, it is IMPLEMENTATION DEFINED whether any accesses from EL0 are trapped to EL2.
 - If HCR_EL2.TIDCP is 0, any accesses from EL0 are UNDEFINED and generate an exception that is taken to EL1 or EL2.

TIDCP	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 accesses to or execution of the specified encodings reserved for IMPLEMENTATION DEFINED functionality are trapped to EL2, when EL2 is enabled in the current Security state.

An implementation can also include IMPLEMENTATION DEFINED registers that provide additional controls, to give finer-grained control of the trapping of IMPLEMENTATION DEFINED features.

Note

The trapping of accesses to these registers from EL1 is higher priority than an exception resulting from the register access being UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TSC, bit [19]

Trap SMC instructions. Traps EL1 execution of SMC instructions to EL2, when EL2 is enabled in the current Security state.

If execution is in AArch64 state, the trap is reported using EC syndrome value 0x17.

If execution is in AArch32 state, the trap is reported using EC syndrome value 0x13.

Note

HCR_EL2.TSC traps execution of the SMC instruction. It is not a routing control for the SMC exception. Trap exceptions and SMC exceptions have different preferred return addresses.

TSC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	<p>If EL3 is implemented, then any attempt to execute an SMC instruction at EL1 is trapped to EL2, when EL2 is enabled in the current Security state, regardless of the value of SCR_EL3.SMD.</p> <p>If EL3 is not implemented, FEAT_NV is implemented, and HCR_EL2.NV is 1, then any attempt to execute an SMC instruction at EL1 using AArch64 is trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>If EL3 is not implemented, and either FEAT_NV is not implemented or HCR_EL2.NV is 0, then it is IMPLEMENTATION DEFINED whether:</p> <ul style="list-style-type: none"> Any attempt to execute an SMC instruction at EL1 is trapped to EL2, when EL2 is enabled in the current Security state. Any attempt to execute an SMC instruction is UNDEFINED.

In AArch32 state, the Armv8-A architecture permits, but does not require, this trap to apply to conditional SMC instructions that fail their condition code check, in the same way as with traps on other conditional instructions.

SMC instructions are UNDEFINED at EL0.

If EL3 is not implemented, and either FEAT_NV is not implemented or HCR_EL2.NV is 0, then it is IMPLEMENTATION DEFINED whether this bit is:

- RES0.
- Implemented with the functionality as described in HCR_EL2.TSC.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID3, bit [18]

Trap ID group 3. Traps EL1 reads of group 3 ID registers to EL2, when EL2 is enabled in the current Security state, as follows:

In AArch64 state:

- Reads of the following registers are trapped to EL2, reported using EC syndrome value 0x18:
 - [ID_PFR0_EL1](#), [ID_PFR1_EL1](#), [ID_PFR2_EL1](#), [ID_DFR0_EL1](#), [ID_AFR0_EL1](#), [ID_MMFR0_EL1](#), [ID_MMFR1_EL1](#), [ID_MMFR2_EL1](#), [ID_MMFR3_EL1](#), [ID_ISAR0_EL1](#), [ID_ISAR1_EL1](#), [ID_ISAR2_EL1](#), [ID_ISAR3_EL1](#), [ID_ISAR4_EL1](#), [ID_ISAR5_EL1](#), [MVFR0_EL1](#), [MVFR1_EL1](#), [MVFR2_EL1](#).
 - [ID_AA64PFR0_EL1](#), [ID_AA64PFR1_EL1](#), [ID_AA64DFR0_EL1](#), [ID_AA64DFR1_EL1](#), [ID_AA64ISAR0_EL1](#), [ID_AA64ISAR1_EL1](#), [ID_AA64MMFR0_EL1](#), [ID_AA64MMFR1_EL1](#), [ID_AA64AFR0_EL1](#), [ID_AA64AFR1_EL1](#).
 - [ID_MMFR4_EL1](#) and [ID_MMFR5_EL1](#) are trapped to EL2.
 - [ID_AA64MMFR2_EL1](#) and [ID_ISAR6_EL1](#) are trapped to EL2.
 - [ID_DFR1_EL1](#) is trapped to EL2.
 - [ID_AA64ZFR0_EL1](#) is trapped to EL2.
 - [ID_AA64SMFR0_EL1](#) is trapped to EL2.
 - [ID_AA64ISAR2_EL1](#) is trapped to EL2.
 - This field traps all MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.

ID_AA64MMFR3_EL1.**If FEAT_FGT is implemented:**

- - **ID_MMFR4_EL1** and **ID_MMFR5_EL1** are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_MMFR4_EL1** or **ID_MMFR5_EL1** are trapped to EL2.
 - **ID_AA64MMFR2_EL1** and **ID_ISAR6_EL1** are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64MMFR2_EL1** or **ID_ISAR6_EL1** are trapped to EL2.
 - **ID_DFR1_EL1** is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_DFR1_EL1** are trapped to EL2.
 - **ID_AA64ZFR0_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64ZFR0_EL1** are trapped to EL2.
 - **ID_AA64SMFR0_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64SMFR0_EL1** are trapped to EL2.
 - **ID_AA64ISAR2_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64ISAR2_EL1** are trapped to EL2.
 - Otherwise, it is IMPLEMENTATION DEFINED whether this bit traps MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.

ID_AA64PFR2_EL1.**If FEAT_FGT is not implemented:**

- **If FEAT_FGT is implemented:**
 - **ID_MMFR4_EL1** and **ID_MMFR5_EL1** are trapped to EL2.
 - **ID_AA64MMFR2_EL1** and **ID_ISAR6_EL1** are trapped to EL2.
 - **ID_DFR1_EL1** is trapped to EL2.
 - **ID_AA64ZFR0_EL1** is trapped to EL2.
 - **ID_AA64SMFR0_EL1** is trapped to EL2.
 - **ID_AA64ISAR2_EL1** is trapped to EL2.
 - This field traps all MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.
- **If FEAT_FGT is not implemented:**
 - **ID_MMFR4_EL1** and **ID_MMFR5_EL1** are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_MMFR4_EL1** or **ID_MMFR5_EL1** are trapped to EL2.
 - **ID_AA64MMFR2_EL1** and **ID_ISAR6_EL1** are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64MMFR2_EL1** or **ID_ISAR6_EL1** are trapped to EL2.
 - **ID_DFR1_EL1** is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to **ID_DFR1_EL1** are trapped to EL2.
 - **ID_AA64ZFR0_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64ZFR0_EL1** are trapped to EL2.
 - **ID_AA64SMFR0_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64SMFR0_EL1** are trapped to EL2.
 - **ID_AA64ISAR2_EL1** is trapped to EL2, unless implemented as RAZ then it is IMPLEMENTATION DEFINED whether accesses to **ID_AA64ISAR2_EL1** are trapped to EL2.

- Otherwise, it is IMPLEMENTATION DEFINED whether this bit traps MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c1-c7}, op2 == {0-7}.

In AArch32 state:

- VMRS access to [MVFR0](#), [MVFR1](#), and [MVFR2](#), are trapped to EL2, reported using EC syndrome value 0x08, unless access is also trapped by [HCPTR](#) which takes priority.
- MRC access to the following registers are trapped to EL2, reported using EC syndrome value 0x03:
 - [ID_PFR0](#), [ID_PFR1](#), [ID_PFR2](#), [ID_DFR0](#), [ID_AFR0](#), [ID_MMFR0](#), [ID_MMFR1](#), [ID_MMFR2](#), [ID_MMFR3](#), [ID_ISAR0](#), [ID_ISAR1](#), [ID_ISAR2](#), [ID_ISAR3](#), [ID_ISAR4](#), [ID_ISAR5](#).
 - If FEAT_FGT is implemented:
 - [ID_MMFR4](#) and [ID_MMFR5](#) are trapped to EL2.
 - [ID_ISAR6](#) is trapped to EL2.
 - [ID_DFR1](#) is trapped to EL2.
 - This field traps all MRC accesses to encodings in the following range that are not already mentioned in this field description: coproc == p15, opc1 == 0, CRn == c0, CRm == {c2-c7}, opc2 == {0-7}.
 - If FEAT_FGT is not implemented:
 - [ID_MMFR4](#) and [ID_MMFR5](#) are trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to [ID_MMFR4](#) or [ID_MMFR5](#) are trapped.
 - [ID_ISAR6](#) is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to [ID_ISAR6](#) are trapped to EL2.
 - [ID_DFR1](#) is trapped to EL2, unless implemented as RAZ, when it is IMPLEMENTATION DEFINED whether accesses to [ID_DFR1](#) are trapped to EL2.
 - Otherwise, it is IMPLEMENTATION DEFINED whether this bit traps all MRC accesses to registers in the following range not already mentioned in this field description with coproc == p15, opc1 == 0, CRn == c0, CRm == {c2-c7}, opc2 == {0-7}.

TID3	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 read accesses to ID group 3 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID2, bit [17]

Trap ID group 2. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state, as follows:

- If EL1 is using AArch64, reads of [CTR_EL0](#), [CCSIDR_EL1](#), [CCSIDR2_EL1](#), [CLIDR_EL1](#), and [CSSELR_EL1](#) are trapped to EL2, reported using EC syndrome value 0x18.
- If EL0 is using AArch64 and the value of [SCTLR_EL1](#).UCT is not 0, reads of [CTR_EL0](#) are trapped to EL2, reported using EC syndrome value 0x18. If the value of [SCTLR_EL1](#).UCT is 0, then EL0 reads of [CTR_EL0](#) are trapped to EL1 and the resulting exception takes precedence over this trap.
- If EL1 is using AArch64, writes to [CSSELR_EL1](#) are trapped to EL2, reported using EC syndrome value 0x18.
- If EL1 is using AArch32, reads of [CTR](#), [CCSIDR](#), [CCSIDR2](#), [CLIDR](#), and [CSSELR](#) are trapped to EL2, reported using EC syndrome value 0x03.
- If EL1 is using AArch32, writes to [CSSELR](#) are trapped to EL2, reported using EC syndrome value 0x03.

TID2	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 and EL0 accesses to ID group 2 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID1, bit [16]

Trap ID group 1. Traps EL1 reads of the following registers to EL2, when EL2 is enabled in the current Security state as follows:

- In AArch64 state, accesses of [REVIDR_EL1](#), [AIDR_EL1](#), [SMIDR_EL1](#), reported using EC syndrome value 0x18.
- In AArch32 state, accesses of [TCMTR](#), [TLBTR](#), [REVIDR](#), [AIDR](#), reported using EC syndrome value 0x03.

TID1	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 read accesses to ID group 1 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TID0, bit [15]

When AArch32 is supported:

Trap ID group 0. Traps the following register accesses to EL2:

- EL1 reads of the [JIDR](#), reported using EC syndrome value 0x05.
- If the [JIDR](#) is RAZ from EL0, EL0 reads of the [JIDR](#), reported using EC syndrome value 0x05.
- EL1 accesses using VMRS of the [FPSID](#), reported using EC syndrome value 0x08.

Note

- It is IMPLEMENTATION DEFINED whether the [JIDR](#) is RAZ or UNDEFINED at EL0. If it is UNDEFINED at EL0, then any resulting exception takes precedence over this trap.
- The [FPSID](#) is not accessible at EL0 using AArch32.
- Writes to the [FPSID](#) are ignored, and not trapped by this control.

TID0	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified EL1 read accesses to ID group 0 registers are trapped to EL2, when EL2 is enabled in the current Security state.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWE, bit [14]

Traps EL0 and EL1 execution of WFE instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFET instruction.

TWE	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFE instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWE or SCTLR_EL1.nTWE .

In AArch32 state, the attempted execution of a conditional WFE instruction is trapped only if the instruction passes its condition code check.

Note

Since a WFE can complete at any time, even without a Wakeup event, the traps on WFE are not guaranteed to be taken, even if the WFE is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information about when WFE instructions can cause the PE to enter a low-power state, see 'Wait for Event mechanism and Send event'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TWI, bit [13]

Traps EL0 and EL1 execution of WFI instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFIT instruction.

TWI	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFI instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWI or SCTLR_EL1.nTWI .

In AArch32 state, the attempted execution of a conditional WFI instruction is trapped only if the instruction passes its condition code check.

Note

Since a WFI can complete at any time, even without a Wakeup event, the traps on WFI are not guaranteed to be taken, even if the WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction

does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information about when WFI instructions can cause the PE to enter a low-power state, see 'Wait for Interrupt'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DC, bit [12]

Default Cacheability.

DC	Meaning
0b0	This control has no effect on the EL1&0 translation regime.
0b1	In any Security state: <ul style="list-style-type: none"> • When EL1 is using AArch64, the PE behaves as if the value of the SCTLR_EL1.M field is 0 for all purposes other than returning the value of a direct read of SCTLR_EL1. • When EL1 is using AArch32, the PE behaves as if the value of the SCTLR.M field is 0 for all purposes other than returning the value of a direct read of SCTLR. • The PE behaves as if the value of the HCR_EL2.VM field is 1 for all purposes other than returning the value of a direct read of HCR_EL2. • The memory type produced by stage 1 of the EL1&0 translation regime is Normal Non-Shareable, Inner Write-Back Read-Allocate Write-Allocate, Outer Write-Back Read-Allocate Write-Allocate.

This field has no effect on the EL2, EL2&0, and EL3 translation regimes.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BSU, bits [11:10]

Barrier Shareability upgrade. This field determines the minimum shareability domain that is applied to any barrier instruction executed from EL1 or EL0:

BSU	Meaning
0b00	No effect.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Full system.

This value is combined with the specified level of the barrier held in its instruction, using the same principles as combining the shareability attributes from two stages of address translation.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0b00 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FB, bit [9]

Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from EL1:

AArch32: [BPIALL](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [TLBIMVAA](#), [ICIALLU](#), [TLBIMVAL](#), [TLBIMVAAL](#).

AArch64: [TLBI VMALLE1](#), [TLBI VAE1](#), [TLBI ASIDE1](#), [TLBI VAAE1](#), [TLBI VALE1](#), [TLBI VAALE1](#), [IC IALLU](#), [TLBI RVAE1](#), [TLBI RVAAE1](#), [TLBI RVALE1](#), [TLBI RVAALE1](#).

FB	Meaning
0b0	This field has no effect on the operation of the specified instructions.
0b1	When one of the specified instruction is executed at EL1, the instruction is broadcast within the Inner Shareable shareability domain.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VSE, bit [8]

Virtual SError interrupt.

VSE	Meaning
0b0	This mechanism is not making a virtual SError interrupt pending.
0b1	A virtual SError interrupt is pending because of this mechanism.

The virtual SError interrupt is enabled only when the value of HCR_EL2.{TGE, AMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VI, bit [7]

Virtual IRQ Interrupt.

VI	Meaning
0b0	This mechanism is not making a virtual IRQ pending.
0b1	A virtual IRQ is pending because of this mechanism.

The virtual IRQ is enabled only when the value of HCR_EL2.{TGE, IMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VF, bit [6]

Virtual FIQ Interrupt.

VF	Meaning
0b0	This mechanism is not making a virtual FIQ pending.
0b1	A virtual FIQ is pending because of this mechanism.

The virtual FIQ is enabled only when the value of HCR_EL2.{TGE, FMO} is {0, 1}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AMO, bit [5]

Physical SError interrupt routing.

AMO	Meaning
0b0	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical SError interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical SError interrupts are taken to EL2 unless they are routed to EL3. Virtual SError interrupts are disabled.
0b1	When executing at any Exception level, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> Physical SError interrupts are taken to EL2, unless they are routed to EL3. When the value of HCR_EL2.TGE is 0, then virtual SError interrupts are enabled.

If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the AMO bit physical asynchronous External aborts and SError interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMO, bit [4]

Physical IRQ Routing.

IMO	Meaning
0b0	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical IRQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical IRQ interrupts are taken to EL2 unless they are routed to EL3. Virtual IRQ interrupts are disabled.
0b1	When executing at any Exception level, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> Physical IRQ interrupts are taken to EL2, unless they are routed to EL3. When the value of HCR_EL2.TGE is 0, then Virtual IRQ interrupts are enabled.

If EL2 is enabled in the current Security state, and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the IMO bit, physical IRQ Interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FMO, bit [3]

Physical FIQ Routing.

FMO	Meaning
0b0	When executing at Exception levels below EL2, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical FIQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical FIQ interrupts are taken to EL2 unless they are routed to EL3. Virtual FIQ interrupts are disabled.
0b1	When executing at any Exception level, and EL2 is enabled in the current Security state: <ul style="list-style-type: none"> Physical FIQ interrupts are taken to EL2, unless they are routed to EL3. When HCR_EL2.TGE is 0, then Virtual FIQ interrupts are enabled.

If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:

- Regardless of the value of the FMO bit, physical FIQ Interrupts target EL2 unless they are routed to EL3.
- When FEAT_VHE is not implemented, or if HCR_EL2.E2H is 0, this field behaves as 1 for all purposes other than a direct read of the value of this bit.
- When FEAT_VHE is implemented and HCR_EL2.E2H is 1, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [2]

Protected Table Walk. In the EL1&0 translation regime, a translation table access made as part of a stage 1 translation table walk is subject to a stage 2 translation. The combining of the memory type attributes from the two stages of translation means the access might be made to a type of Device memory. If this occurs, then the value of this bit determines the behavior:

PTW	Meaning
0b0	The translation table walk occurs as if it is to Normal Non-cacheable memory. This means it can be made speculatively.
0b1	The memory access generates a stage 2 Permission fault.

This bit is permitted to be cached in a TLB.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SWIO, bit [1]

Set/Way Invalidation Override. Causes EL1 execution of the data cache invalidate by set/way instructions to perform a data cache clean and invalidate by set/way:

SWIO	Meaning
0b0	This control has no effect on the operation of data cache invalidate by set/way instructions.
0b1	Data cache invalidate by set/way instructions perform a data cache clean and invalidate by set/way.

When the value of this bit is 1:

AArch32: [DCISW](#) performs the same invalidation as a [DCCISW](#) instruction.

AArch64: [DC ISW](#) performs the same invalidation as a [DC CISW](#) instruction.

This bit can be implemented as RES1.

When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VM, bit [0]

Virtualization enable. Enables stage 2 address translation for the EL1&0 translation regime, when EL2 is enabled in the current Security state.

VM	Meaning
0b0	EL1&0 stage 2 address translation disabled.
0b1	EL1&0 stage 2 address translation enabled.

When the value of this bit is 1, data cache invalidate instructions executed at EL1 perform a data cache clean and invalidate. For the invalidate by set/way instruction this behavior applies regardless of the value of the HCR_EL2.SWIO bit.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of HCR_EL2.{E2H, TGE} is {1, 1}, this field behaves as 0 for all purposes other than a direct read of the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing HCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x078];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HCR_EL2;

```

MSR HCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x078] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HCR_EL2 = X[t, 64];
```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

HCRX_EL2, Extended Hypervisor Configuration Register

The HCRX_EL2 characteristics are:

Purpose

Provides configuration controls for virtualization, including defining whether various operations are trapped to EL2.

Configuration

This register is present only when FEAT_HCX is implemented. Otherwise, direct accesses to HCRX_EL2 are UNDEFINED.

If EL2 is not implemented, this register is RES0 from EL3.

The bits in this register behave as if they are 0 for all purposes other than direct reads of the register if:

- EL2 is not enabled in the current Security state.
- [SCR_EL3.HXEn](#) is 0.

Attributes

HCRX_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48																				
RES0										EnIDCP128		MSCE		EnSDERR		MCE2		RES0		CMOW		EnSNERR		VFNM		D128		EnVINM		PTTW		TALLINT		SCT	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																				

Bits [63:22:12]

Reserved, RES0.

EnIDCP128, bit [21]

When FEAT_SYSREG128 is implemented and HCR_EL2.[E2H,TGE] != 0b11:

Enables access to IMPLEMENTATION DEFINED 128-bit System registers.

EnIDCP128	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, accesses at EL1, EL0 to IMPLEMENTATION DEFINED 128-bit System registers are trapped to EL2 using an ESR_EL2.EC value of 0x14, unless the access generates a higher priority exception. Disables the functionality of the 128-bit IMPLEMENTATION DEFINED System registers that are accessible at EL2.
0b1	No accesses are trapped by this control.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnSDERR, bit [20]**When FEAT_ADERR is implemented:**Enable Synchronous Device Read Error. Override [SCTLR2_EL1.EnADERR](#).

EnSDERR	Meaning
0b0	This field has no effect on External aborts on Device memory reads at EL1 and EL0.
0b1	External abort on Device memory reads generate synchronous Data Abort exceptions in the EL1&0 translation regime.

Setting this field to 1 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Device memory read in every case.

Setting this field to 1 might have a performance impact for Device memory reads.

This field is ignored by the PE and treated as zero when any of the following are true:

- [SCR_EL3.HXEn](#) == 0.
- EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [19]

Reserved, RES0.

EnSNERR, bit [18]**When FEAT_ANERR is implemented:**Enable Synchronous Normal Read Error. Override [SCTLR2_EL1.EnANERR](#).

EnSNERR	Meaning
0b0	This field has no effect on External aborts on Normal memory reads at EL1 and EL0.
0b1	External abort on Normal memory reads generate synchronous Data Abort exceptions in the EL1&0 translation regime.

Setting this field to 1 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Normal memory read in every case.

Setting this field to 1 might have a performance impact for Normal memory reads.

This field is ignored by the PE and treated as zero when any of the following are true:

- [SCR_EL3.HXEn](#) == 0.
- EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D128En, bit [17]**When FEAT_D128 is implemented:**

128-bit System Register trap control. Enable access to 128-bit System Registers via MRRS, MSRR instructions.

- If EL1 is using AArch64 state, accesses to the following registers are trapped to EL2 and reported using EC syndrome value 0x14:
 - TTBR0_EL1.
 - TTBR1_EL1.
 - If FEAT_THE is implemented, RCWMASK_EL1, RCWSMASK_EL1.
 - PAR_EL1.

D128En	Meaning
0b0	EL1 accesses to the specified registers are disabled, and trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PTTWI, bit [16]**When FEAT_THE is implemented:**

Permit Translation Table Walk Incoherence.

Permits RCWS instructions to have Reduced Coherence property.

PTTWI	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, write accesses generated by RCWS at EL1&0 do not have the Reduced Coherence property.
0b1	Write accesses generated by RCWS at EL1&0 have the Reduced Coherence property, if enabled by TCR2_EL1.PTTWI.

This bit is permitted to be cached in TLB.

This bit is permitted to be built as a read-only bit with a fixed value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SCTLR2En, bit [15]**When FEAT_SCTLR2 is implemented:**

SCTLR2_EL1 Enable. In AArch64 state, accesses to **SCTLR2_EL1** are trapped to EL2 and reported using EC syndrome value 0x18.

SCTLR2En	Meaning
0b0	Accesses to SCTLR2_EL1 at EL1 are trapped to EL2, unless the access generates a higher priority exception. The value in SCTLR2_EL1 is treated as 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCR2En, bit [14]**When FEAT_TCR2 is implemented:**

TCR2_EL1 Enable. In AArch64 state, accesses to **TCR2_EL1** are trapped to EL2 and reported using EC syndrome value 0x18.

TCR2En	Meaning
0b0	Accesses to TCR2_EL1 at EL1 are trapped to EL2, unless the access generates a higher priority exception. The value in TCR2_EL1 is treated as 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [13:12]

Reserved, RES0.

MSCEn, bit [11]**When FEAT_MOPS is implemented:**

Memory Set and Memory Copy instructions Enable. Enables execution of the CPY*, SETG*, SETP*, SETM*, and SETE* instructions at EL1 or EL0.

MSCEn	Meaning
0b0	Execution of the Memory Copy and Memory Set instructions is UNDEFINED at EL1 or EL0.
0b1	This control does not cause any instructions to be UNDEFINED.

This bit behaves as if it is 1 if any of the following are true:

- EL2 is not implemented or enabled.
- The value of [HCR_EL2](#).{E2H, TGE} is {1, 1}.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MCE2, bit [10]

When FEAT_MOPS is implemented:

Controls Memory Copy and Memory Set exceptions generated as part of attempting to execute the Memory Copy and Memory Set instructions from EL1.

MCE2	Meaning
0b0	Memory Copy and Memory Set exceptions generated from EL1 are taken to EL1.
0b1	Memory Copy and Memory Set exceptions generated from EL1 are taken to EL2.

When the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this control does not affect any exceptions due to the higher priority [SCTLR_EL2](#).MSCEn control.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CMOW, bit [9]

When FEAT_CMOW is implemented:

Controls cache maintenance instruction permission for the following instructions executed at EL1 or EL0.

- [IC IVAU](#), [DC CIVAC](#), [DC CIGDVAC](#) and [DC CIGVAC](#).
- [ICIMVAU](#), [DCCIMVAC](#).

CMOW	Meaning
0b0	These instructions executed at EL1 or EL0 with stage 2 read permission, but without stage 2 write permission do not generate a stage 2 permission fault.
0b1	These instructions executed at EL1 or EL0, if enabled as a result of SCTLR_EL1.UCI=1 , with stage 2 read permission, but without stage 2 write permission generate a stage 2 permission fault.

For this control, stage 2 has write permission if S2AP[1] is 1 or DBM is 1 in the stage 2 descriptor. The instructions do not cause an update to the dirty state.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VFNMI, bit [8]

When FEAT_NMI is implemented:

Virtual FIQ Interrupt with Superpriority. Enables signaling of virtual FIQ interrupts with Superpriority.

VFNMI	Meaning
0b0	When HCR_EL2.VF is 1, a signaled pending virtual FIQ interrupt does not have Superpriority.
0b1	When HCR_EL2.VF is 1, a signaled pending virtual FIQ interrupt has Superpriority.

When [HCR_EL2.VF](#) is 0, this bit has no effect.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

VINMI, bit [7]

When FEAT_NMI is implemented:

Virtual IRQ Interrupt with Superpriority. Enables signaling of virtual IRQ interrupts with Superpriority.

VINMI	Meaning
0b0	When HCR_EL2.VI is 1, a signaled pending virtual IRQ interrupt does not have Superpriority.
0b1	When HCR_EL2.VI is 1, a signaled pending virtual IRQ interrupt has Superpriority.

When [HCR_EL2.VI](#) is 0, this bit has no effect.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

TALLINT, bit [6]**When FEAT_NMI is implemented:**

Trap MSR writes of [ALLINT](#) at EL1 using AArch64 to EL2, when EL2 is implemented and enabled in the current Security state, reported using EC syndrome value 0x18.

TALLINT	Meaning
0b0	MSR writes of ALLINT are not trapped by this mechanism.
0b1	MSR writes of ALLINT at EL1 using AArch64 are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

SMPME, bit [5]**When FEAT_SME is implemented:**

Streaming Mode Priority Mapping Enable.

Controls mapping of the value of [SMPRI_EL1](#). Priority for streaming execution priority at EL0 or EL1.

SMPME	Meaning
0b0	The effective priority value is taken from SMPRI_EL1 .Priority.
0b1	The effective priority value is: <ul style="list-style-type: none"> • When the current Exception level is EL2 or EL3, the value of SMPRI_EL1.Priority. • When the current Exception level is EL0 or EL1, the value of the SMPRIMAP_EL2 field corresponding to the value of SMPRI_EL1.Priority.

When [SMIDR_EL1](#).SMPS is '0', this field is RES0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FGTnXS, bit [4]**When FEAT_XS is implemented:**

Determines if the fine-grained traps in HFGITR_EL2 that apply to each of the TLBI maintenance instructions that are accessible at EL1 also apply to the corresponding TLBI maintenance instructions with the nXS qualifier.

FGTnXS	Meaning
0b0	The fine-grained trap in the HFGITR_EL2 that applies to a TLBI maintenance instruction at EL1 also applies to the corresponding TLBI instruction with the nXS qualifier at EL1.
0b1	The fine-grained trap in the HFGITR_EL2 that applies to a TLBI maintenance instruction at EL1 does not apply to the corresponding TLBI instruction with the nXS qualifier at EL1.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

FnXS, bit [3]**When FEAT_XS is implemented:**

Determines the behavior of TLBI instructions affected by the XS attribute.

This control bit also determines whether an AArch64 DSB instruction behaves as a DSB instruction with an nXS qualifier when executed at EL0 and EL1.

FnXS	Meaning
0b0	This control does not have any effect on the behavior of the TLBI maintenance instructions.
0b1	A TLBI maintenance instruction without the nXS qualifier executed at EL1 behaves in the same way as the corresponding TLBI maintenance instruction with the nXS qualifier. An AArch64 DSB instruction executed at EL1 or EL0 behaves in the same way as the corresponding DSB instruction with the nXS qualifier executed at EL1 or EL0.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

EnASR, bit [2]**When FEAT_LS64_V is implemented:**

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an ST64BV instruction at EL0 or EL1 to EL2.

EnASR	Meaning
0b0	Execution of an ST64BV instruction at EL0 is trapped to EL2 if the execution is not trapped by SCTLR_EL1 .EnASR. Execution of an ST64BV instruction at EL1 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000000.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

EnALS, bit [1]**When FEAT_LS64 is implemented:**

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an LD64B or ST64B instruction at EL0 or EL1 to EL2.

EnALS	Meaning
0b0	Execution of an LD64B or ST64B instruction at EL0 is trapped to EL2 if the execution is not trapped by SCTLR_EL1 .EnALS. Execution of an LD64B or ST64B instruction at EL1 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an LD64B or ST64B instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000002.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

EnAS0, bit [0]**When FEAT_LS64_ACCDATA is implemented:**

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an ST64BV0 instruction at EL0 or EL1 to EL2.

EnAS0	Meaning
0b0	Execution of an ST64BV0 instruction at EL0 is trapped to EL2 if the execution is not trapped by SCTLR_EL1.EnAS0 . Execution of an ST64BV0 instruction at EL1 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV0 instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000001.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when EL3 is not implemented and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

Accessing HCRX_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HCRX_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCRX_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0xA0];
    elsif EL2Enabled() && HCRX_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.HXEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.HXEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HCRX_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HCRX_EL2;

```

MSR HCRX_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0xA0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.HXEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.HXEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HCRX_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HCRX_EL2 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

HDFGRTR2_EL2, Hypervisor Debug Fine-Grained Read Trap Register 2

The HDFGRTR2_EL2 characteristics are:

Purpose

Provides controls for traps of MRS and MRC reads of debug, trace, PMU, and Statistical Profiling System registers.

Configuration

This register is present only when FEAT_FGT2 is implemented. Otherwise, direct accesses to HDFGRTR2_EL2 are UNDEFINED.

Attributes

HDFGRTR2_EL2 is a 64-bit register.

Field descriptions

6362616059585756555453	52	51	50	49	48	47
RES0	nTRCITECR_EL1	nPMDSFR_EL1	nSPMDEVAFF_EL1	nSPMID	nSPMSCR_EL1	nSPMACCESSR_EL1
3130292827262524232221	20	19	18	17	16	15

Bits [63:21]

Reserved, RES0.

nTRCITECR_EL1, bit [20] When FEAT_ITE is implemented:

Trap MRS reads of TRCITECR_EL1 at EL1 using AArch64 to EL2.

nTRCITECR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of TRCITECR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of TRCITECR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMDSFR_EL1, bit [19]**When FEAT_SPE_FDS is implemented:**

Trap MRS reads of [PMDSFR_EL1](#) at EL1 using AArch64 to EL2.

nPMDSFR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of PMDSFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMDSFR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMDEVAFF_EL1, bit [18]**When FEAT_SPMU is implemented:**

Trap MRS reads of [SPMDEVAFF_EL1](#) at EL1 using AArch64 to EL2.

nSPMDEVAFF_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of SPMDEVAFF_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SPMDEVAFF_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMID, bit [17]**When FEAT_SPMU is implemented:**

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMCEGR_EL1](#).
- [SPMCGCR<n>_EL1](#).
- [SPMDEVARCH_EL1](#).
- [SPMIIDR_EL1](#).

nSPMID	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMSCR_EL1, bit [16]**When FEAT_SPMU is implemented:**

Trap MRS reads of [SPMSCR_EL1](#) at EL1 using AArch64 to EL2.

nSPMSCR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of SPMSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SPMSCR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMACCESSR_EL1, bit [15]**When FEAT_SPMU is implemented:**

Trap MRS reads of [SPMACCESSR_EL1](#) at EL1 using AArch64 to EL2.

nSPMACCESSR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of SPMACCESSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SPMACCESSR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMCR_EL0, bit [14]

When FEAT_SPMU is implemented:

Trap MRS reads of [SPMCR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nSPMCR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of SPMCR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SPMCR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMOVS, bit [13]

When FEAT_SPMU is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMOVSLR_EL0](#).
- [SPMOVSSET_EL0](#).

nSPMOVS	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMINTEN, bit [12]

When FEAT_SPMU is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMINTENCLR_EL1](#).
- [SPMINTENSET_EL1](#).

nSPMINTEN	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMCNTEN, bit [11]

When FEAT_SPMU is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMCNTENCLR_EL0](#).
- [SPMCNTENSET_EL0](#).

nSPMCNTEN	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMSELR_EL0, bit [10]

When FEAT_SPMU is implemented:

Trap MRS reads of [SPMSELR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nSPMSELR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads of SPMSELR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SPMSELR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMEVTYPEPn_EL0, bit [9]

When FEAT_SPMU is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMEVTYPEP<n>_EL0](#).
- [SPMEVFILTR<n>_EL0](#).
- [SPMEVFILT2R<n>_EL0](#).

nSPMEVTYPE_{Rn}_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEN2 == 1, then MRS reads at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

Regardless of the value of this bit, if event counter n is not implemented, a read of [SPMEVTYPE_{R<n>_EL0}](#), [SPMEVFILTR<n>_EL0](#), or [SPMEVFILT2R<n>_EL0](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMEVCNTR_n_EL0, bit [8]

When FEAT_SPMU is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2: [SPMEVCNTR<n>_EL0](#).

nSPMEVCNTR_n_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEN2 == 1, then MRS reads at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

Regardless of the value of this bit, if event counter n is not implemented, a read of [SPMEVCNTR<n>_EL0](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMSSCR_EL1, bit [7]

When FEAT_PMuV3_SS is implemented:

Trap MRS reads of [PMSSCR_EL1](#) at EL1 using AArch64 to EL2.

nPMSSCR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of PMSSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMSSCR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMSSDATA, bit [6]

When FEAT_PMuV3_SS is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [PMCCNTSVR_EL1](#).
- [PMEVCNTSVR<n>_EL1](#).
- [PMICNTSVR_EL1](#), if FEAT_PMuV3_ICNTR is implemented.

nPMSSDATA	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nMDSELR_EL1, bit [5]

When FEAT_Debugv8p9 is implemented:

Trap MRS reads of [MDSELR_EL1](#) at EL1 using AArch64 to EL2.

nMDSELR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of MDSELR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of MDSELR_EL1 are not trapped by this mechanism.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and [MDSELR_EL1](#) is implemented as RAZ/WI.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMUACR_EL1, bit [4]

When FEAT_PMuV3p9 is implemented:

Trap MRS reads of [PMUACR_EL1](#) at EL1 using AArch64 to EL2.

nPMUACR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of PMUACR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMUACR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMICFILTR_EL0, bit [3]

When FEAT_PMuV3_ICNTR is implemented:

Trap MRS reads of [PMICFILTR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nPMICFILTR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTE _{n2} == 1, then MRS reads of PMICFILTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMICFILTR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMICNTR_EL0, bit [2]

When FEAT_PMuV3_ICNTR is implemented:

Trap MRS reads of [PMICNTR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nPMICNTR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTE _{n2} == 1, then MRS reads of PMICNTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMICNTR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMIAR_EL1, bit [1]

When FEAT_SEBEP is implemented:

Trap MRS reads of [PMIAR_EL1](#) at EL1 using AArch64 to EL2.

nPMIAR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _{n2} == 1, then MRS reads of PMIAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMIAR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMECR_EL1, bit [0]

When FEAT_EBEP is implemented:

Trap MRS reads of [PMECR_EL1](#) at EL1 using AArch64 to EL2.

nPMECR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS reads of PMECR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMECR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing HDFGRTR2_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HDFGRTR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1A0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HDFGRTR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HDFGRTR2_EL2;

```

MSR HDFGRTR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1A0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HDFGRTR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HDFGRTR2_EL2 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HDFGRTR_EL2, Hypervisor Debug Fine-Grained Read Trap Register

The HDFGRTR_EL2 characteristics are:

Purpose

Provides controls for traps of MRS and MRC reads of debug, trace, PMU, and Statistical Profiling System registers.

Configuration

This register is present only when FEAT_FGT is implemented. Otherwise, direct accesses to HDFGRTR_EL2 are UNDEFINED.

Attributes

HDFGRTR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56
PMBIDR_EL1	nPMSNEVFR_EL1	nBRBDATA	nBRBCTL	nBRBIDR	PMCEIDn_EL0	PMUSERENR_EL0	TRBTRG_EL1
PMSIRR_EL1	PMSIDR_EL1	PMSICR_EL1	PMSFCR_EL1	PMSEVFR_EL1	PMSCR_EL1	PMBSR_EL1	PMBPTR_EL1
31	30	29	28	27	26	25	24

PMBIDR_EL1, bit [63] **When FEAT_SPE is implemented:**

Trap MRS reads of [PMBIDR_EL1](#) at EL1 using AArch64 to EL2.

PMBIDR_EL1	Meaning
0b0	MRS reads of PMBIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMBIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPMSNEVFR_EL1, bit [62] **When FEAT_SPEv1p2 is implemented:**

Trap MRS reads of [PMSNEVFR_EL1](#) at EL1 using AArch64 to EL2.

nPMSNEVFR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSNEVFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PMSNEVFR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBDATA, bit [61]

When FEAT_BRBE is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [BRBINF<n>_EL1](#).
- [BRBINFINJ_EL1](#).
- [BRBSRC<n>_EL1](#).
- [BRBSRCINJ_EL1](#).
- [BRBTGT<n>_EL1](#).
- [BRBTGTINJ_EL1](#).
- [BRBTS_EL1](#).

nBRBDATA	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBCTL, bit [60]

When FEAT_BRBE is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [BRBCR_EL1](#).
- [BRBFCR_EL1](#).

nBRBCTL	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBIDR, bit [59]

When FEAT_BRBE is implemented:

Trap MRS reads of [BRBIDR0_EL1](#) at EL1 using AArch64 to EL2.

nBRBIDR	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of BRBIDR0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of BRBIDR0_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCEIDn_EL0, bit [58]

When FEAT_PMUv3 is implemented:

Trap MRS reads of PMCEID<n>_EL0 at EL1 and EL0 using AArch64 and MRC reads of PMCEID<n> at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCEIDn_EL0	Meaning
0b0	MRS reads of PMCEID<n>_EL0 at EL1 and EL0 using AArch64 and MRC reads of PMCEID<n> at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> MRS reads of PMCEID<n>_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads of PMCEID<n> at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMUSERENR_EL0, bit [57]

When FEAT_PMuV3 is implemented:

Trap MRS reads of [PMUSERENR_EL0](#) at EL1 and EL0 using AArch64 and MRC reads of [PMUSERENR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMUSERENR_EL0	Meaning
0b0	MRS reads of PMUSERENR_EL0 at EL1 and EL0 using AArch64 and MRC reads of PMUSERENR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> MRS reads of PMUSERENR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads of PMUSERENR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBTRG_EL1, bit [56]

When FEAT_TRBE is implemented:

Trap MRS reads of [TRBTRG_EL1](#) at EL1 using AArch64 to EL2.

TRBTRG_EL1	Meaning
0b0	MRS reads of TRBTRG_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBTRG_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBSR_EL1, bit [55]

When FEAT_TRBE is implemented:

Trap MRS reads of [TRBSR_EL1](#) at EL1 using AArch64 to EL2.

TRBSR_EL1	Meaning
0b0	MRS reads of TRBSR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBPTR_EL1, bit [54]

When FEAT_TRBE is implemented:

Trap MRS reads of [TRBPTR_EL1](#) at EL1 using AArch64 to EL2.

TRBPTR_EL1	Meaning
0b0	MRS reads of TRBPTR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBPTR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBMAR_EL1, bit [53]**When FEAT_TRBE is implemented:**

Trap MRS reads of [TRBMAR_EL1](#) at EL1 using AArch64 to EL2.

TRBMAR_EL1	Meaning
0b0	MRS reads of TRBMAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBMAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBLIMITR_EL1, bit [52]**When FEAT_TRBE is implemented:**

Trap MRS reads of [TRBLIMITR_EL1](#) at EL1 using AArch64 to EL2.

TRBLIMITR_EL1	Meaning
0b0	MRS reads of TRBLIMITR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBLIMITR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBIDR_EL1, bit [51]**When FEAT_TRBE is implemented:**

Trap MRS reads of [TRBIDR_EL1](#) at EL1 using AArch64 to EL2.

TRBIDR_EL1	Meaning
0b0	MRS reads of TRBIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBBASER_EL1, bit [50]

When FEAT_TRBE is implemented:

Trap MRS reads of [TRBBASER_EL1](#) at EL1 using AArch64 to EL2.

TRBBASER_EL1	Meaning
0b0	MRS reads of TRBBASER_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRBBASER_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [49]

Reserved, RES0.

TRCVICTLR, bit [48]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCVICTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCVICTLR at EL1 using AArch64 to EL2.

TRCVICTLR	Meaning
0b0	MRS reads of TRCVICTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCVICTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCSTATR, bit [47]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCSTATR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCSTATR at EL1 using AArch64 to EL2.

TRCSTATR	Meaning
0b0	MRS reads of TRCSTATR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCSTATR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCSSCSRn, bit [46]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCSSCSR<n> are implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCSSCSR<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCSSCSR<n> at EL1 using AArch64 to EL2.

TRCSSCSRn	Meaning
0b0	MRS reads of TRCSSCSR<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCSSCSR<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If Single-shot Comparator n is not implemented, a read of [TRCSSCSR<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCSEQSTR, bit [45]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCSEQSTR is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCSEQSTR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCSEQSTR at EL1 using AArch64 to EL2.

TRCSEQSTR	Meaning
0b0	MRS reads of TRCSEQSTR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCSEQSTR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCPRGCTLR, bit [44]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCPRGCTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCPRGCTLR at EL1 using AArch64 to EL2.

TRCPRGCTLR	Meaning
0b0	MRS reads of TRCPRGCTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCPRGCTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCOSLSR, bit [43]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCOSLSR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCOSLSR at EL1 using AArch64 to EL2.

TRCOSLSR	Meaning
0b0	MRS reads of TRCOSLSR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCOSLSR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [42]

Reserved, RES0.

TRCIMSPECn, bit [41]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCIMSPEC<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCIMSPEC<n> at EL1 using AArch64 to EL2.

TRCIMSPECn	Meaning
0b0	MRS reads of TRCIMSPEC<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCIMSPEC<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

TRCIMSPEC<1-7> are optional. If [TRCIMSPEC<n>](#) is not implemented, a read of [TRCIMSPEC<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCID, bit [40]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- In an Armv9 implementation:
 - [TRCDEVARCH](#).
 - [TRCDEVID](#).
 - All of the TRCIDR<n> registers.
- In an Armv8 implementation:
 - ETM TRCDEVARCH.
 - ETM TRCDEVID.
 - All of the ETM TRCIDR<n> registers.

TRCID	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [39:38]

Reserved, RES0.

TRCCNTVRn, bit [37]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCCNTVR<n> are implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCCNTVR<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCCNTVR<n> at EL1 using AArch64 to EL2.

TRCCNTVRn	Meaning
0b0	MRS reads of TRCCNTVR<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCCNTVR<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If Counter n is not implemented, a read of [TRCCNTVR<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCCLAIM, bit [36]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCCLAIMCLR](#) and [TRCCLAIMSET](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCCLAIMCLR and ETM TRCCLAIMSET at EL1 using AArch64 to EL2.

TRCCLAIM	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCAUXCTLR, bit [35]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCAUXCTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MRS reads of ETM TRCAUXCTLR at EL1 using AArch64 to EL2.

TRCAUXCTLR	Meaning
0b0	MRS reads of TRCAUXCTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCAUXCTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCAUTHSTATUS, bit [34]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of [TRCAUTHSTATUS](#) at EL1 using AArch64 to EL2.

TRCAUTHSTATUS	Meaning
0b0	MRS reads of TRCAUTHSTATUS are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TRCAUTHSTATUS at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRC, bit [33]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MRS reads of the following registers at EL1 using AArch64 to EL2:

- [TRCACATR<n>](#).
- [TRCACVR<n>](#).
- [TRCBBCTLR](#).
- [TRCCCCTLR](#).
- [TRCCIDCCTLR0](#).
- [TRCCIDCCTLR1](#).
- [TRCCIDCVR<n>](#).
- [TRCCNTCTLR<n>](#).
- [TRCCNTRLDVR<n>](#).
- [TRCCONFIGR](#).
- [TRCEVENTCTL0R](#).
- [TRCEVENTCTL1R](#).
- [TRCEXTINSELR<n>](#).
- [TRCOCTLR](#).
- [TRCRSCTLR<n>](#).
- [TRCRSR](#).
- [TRCSEQEVR<n>](#).
- [TRCSEQRSTEVR](#).
- [TRCSSCCR<n>](#).
- [TRCSSPCICR<n>](#).
- [TRCSTALLCTLR](#).
- [TRCSYNCPR](#).
- [TRCTRACEIDR](#).
- [TRCTSCTLR](#).
- [TRCVIIECTLR](#).
- [TRCVIPCSSCTLR](#).
- [TRCVISSCTLR](#).
- [TRCVMIDCCTLR0](#).
- [TRCVMIDCCTLR1](#).
- [TRCVMIDCVR<n>](#).

In an Armv8 implementation, trap MRS reads of the following registers at EL1 using AArch64 to EL2:

- ETM [TRCACATR<n>](#).

- ETM TRCACVR<n>.
- ETM TRCBBCTLR.
- ETM TRCCCCTLR.
- ETM TRCCIDCCTLR0.
- ETM TRCCIDCCTLR1.
- ETM TRCCIDCVR<n>.
- ETM TRCCNTCTLR<n>.
- ETM TRCCNTRLDVR<n>.
- ETM TRCCONFIGR.
- ETM TRCEVENTCTL0R.
- ETM TRCEVENTCTL1R.
- ETM TRCEXTINSELR.
- ETM TRCQCTLR.
- ETM TRCRSCTLR<n>.
- ETM TRCSEQEVR<n>.
- ETM TRCSEQRSTEVR.
- ETM TRCSSCCR<n>.
- ETM TRCSSPCICR<n>.
- ETM TRCSTALLCTLR.
- ETM TRCSYNCPR.
- ETM TRCTRACEIDR.
- ETM TRCTSCTLR.
- ETM TRCVIIECTLR.
- ETM TRCVIPCSSCTLR.
- ETM TRCVISSCTLR.
- ETM TRCVMIDCCTLR0.
- ETM TRCVMIDCCTLR1.
- ETM TRCVMIDCVR<n>.

TRC	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

A read of an unimplemented register is UNDEFINED.

[TRCEXTINSELR<n>](#) and [TRCRSR](#) are only implemented if FEAT_ETE is implemented.

TRCEXTINSELR is only implemented if FEAT_ETE is not implemented and FEAT_ETMv4 is implemented.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSLATFR_EL1, bit [32]

When FEAT_SPE is implemented:

Trap MRS reads of [PMSLATFR_EL1](#) at EL1 using AArch64 to EL2.

PMSLATFR_EL1	Meaning
0b0	MRS reads of PMSLATFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSLATFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSIRR_EL1, bit [31]

When FEAT_SPE is implemented:

Trap MRS reads of [PMSIRR_EL1](#) at EL1 using AArch64 to EL2.

PMSIRR_EL1	Meaning
0b0	MRS reads of PMSIRR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSIRR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSIDR_EL1, bit [30]

When FEAT_SPE is implemented:

Trap MRS reads of [PMSIDR_EL1](#) at EL1 using AArch64 to EL2.

PMSIDR_EL1	Meaning
0b0	MRS reads of PMSIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSICR_EL1, bit [29]**When FEAT_SPE is implemented:**

Trap MRS reads of [PMSICR_EL1](#) at EL1 using AArch64 to EL2.

PMSICR_EL1	Meaning
0b0	MRS reads of PMSICR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSICR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSFCR_EL1, bit [28]**When FEAT_SPE is implemented:**

Trap MRS reads of [PMSFCR_EL1](#) at EL1 using AArch64 to EL2.

PMSFCR_EL1	Meaning
0b0	MRS reads of PMSFCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSFCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSEVFR_EL1, bit [27]**When FEAT_SPE is implemented:**

Trap MRS reads of [PMSEVFR_EL1](#) at EL1 using AArch64 to EL2.

PMSEVFR_EL1	Meaning
0b0	MRS reads of PMSEVFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSEVFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSCR_EL1, bit [26]

When FEAT_SPE is implemented:

Trap MRS reads of [PMSCR_EL1](#) at EL1 using AArch64 to EL2.

PMSCR_EL1	Meaning
0b0	MRS reads of PMSCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBSR_EL1, bit [25]

When FEAT_SPE is implemented:

Trap MRS reads of [PMBSR_EL1](#) at EL1 using AArch64 to EL2.

PMBSR_EL1	Meaning
0b0	MRS reads of PMBSR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMBSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBPTR_EL1, bit [24]**When FEAT_SPE is implemented:**

Trap MRS reads of [PMBPTR_EL1](#) at EL1 using AArch64 to EL2.

PMBPTR_EL1	Meaning
0b0	MRS reads of PMBPTR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMBPTR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBLIMITR_EL1, bit [23]**When FEAT_SPE is implemented:**

Trap MRS reads of [PMBLIMITR_EL1](#) at EL1 using AArch64 to EL2.

PMBLIMITR_EL1	Meaning
0b0	MRS reads of PMBLIMITR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PMBLIMITR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMMIR_EL1, bit [22]**When FEAT_PMUv3 is implemented:**

Trap MRS reads of [PMMIR_EL1](#) at EL1 using AArch64 to EL2.

PMMIR_EL1	Meaning
0b0	MRS reads of PMMIR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MRS reads of PMMIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [21:20]

Reserved, RES0.

PMSELR_ELO, bit [19]

When FEAT_PMUv3 is implemented:

Trap MRS reads of [PMSELR_ELO](#) at EL1 and EL0 using AArch64 and MRC reads of [PMSELR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMSELR_ELO	Meaning
0b0	MRS reads of PMSELR_ELO at EL1 and EL0 using AArch64 and MRC reads of PMSELR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> MRS reads of PMSELR_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads of PMSELR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMOVS, bit [18]

When FEAT_PMUv3 is implemented:

Trap MRS reads and MRC reads of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MRS reads of [PMOVSCLR_ELO](#) and [PMOVSSET_ELO](#).
- At EL0 using AArch32 when EL1 is using AArch64: MRC reads of [PMOVSR](#) and [PMOVSSET](#).

PMOVS	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the read generates a higher priority exception:</p> <ul style="list-style-type: none"> MRS reads at EL1 and EL0 using AArch64 of PMOVSCLR_ELO and PMOVSSSET_ELO are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads at EL0 using AArch32 of PMOVSR and PMOVSSSET are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMINTEN, bit [17]

When FEAT_PMUv3 is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [PMINTENCLR_EL1](#).
- [PMINTENSET_EL1](#).

PMINTEN	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.</p>

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCNTEN, bit [16]

When FEAT_PMUv3 is implemented:

Trap MRS reads and MRC reads of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MRS reads of [PMCNTENCLR_ELO](#) and [PMCNTENSET_ELO](#).
- At EL0 using AArch32 when EL1 is using AArch64: MRC reads of [PMCNTENCLR](#) and [PMCNTENSET](#).

PMCNTEN	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the read generates a higher priority exception:</p> <ul style="list-style-type: none"> MRS reads at EL1 and EL0 using AArch64 of PMCNTENCLR_EL0 and PMCNTENSET_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads at EL0 using AArch32 of PMCNTENCLR and PMCNTENSET are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCCNTR_EL0, bit [15]

When FEAT_PMUv3 is implemented:

Trap MRS reads of [PMCCNTR_EL0](#) at EL1 and EL0 using AArch64 and MRC and MRRC reads of [PMCCNTR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCCNTR_EL0	Meaning
0b0	MRS reads of PMCCNTR_EL0 at EL1 and EL0 using AArch64 and MRC and MRRC reads of PMCCNTR at EL0 using AArch32 are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the read generates a higher priority exception:</p> <ul style="list-style-type: none"> MRS reads of PMCCNTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC and MRRC reads of PMCCNTR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03 (for MRC) or 0x04 (for MRRC).

[PMCCNTR_EL0](#) is indirectly accessed when [PMCR_EL0](#).C is set to 0b1.

Setting this field to 1 has no effect on accesses to [PMCCNTR_EL0](#) using [PMCR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCCFILTR_EL0, bit [14]**When FEAT_PMuV3 is implemented:**

Trap MRS reads of [PMCCFILTR_EL0](#) at EL1 and EL0 using AArch64 and MRC reads of [PMCCFILTR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCCFILTR_EL0	Meaning
0b0	MRS reads of PMCCFILTR_EL0 at EL1 and EL0 using AArch64 and MRC reads of PMCCFILTR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> • MRS reads of PMCCFILTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. • MRC reads of PMCCFILTR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

[PMCCFILTR_EL0](#) can also be accessed in AArch64 state using [PMXEVTYPER_EL0](#) when [PMSELR_EL0](#).SEL == 31, and [PMCCFILTR](#) can also be accessed in AArch32 state using [PMXEVTYPER](#) when [PMSELR](#).SEL == 31.

Setting this field to 1 has no effect on accesses to [PMXEVTYPER_EL0](#) and [PMXEVTYPER](#), regardless of the value of [PMSELR_EL0](#).SEL or [PMSELR](#).SEL.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMEVTYPERn_EL0, bit [13]**When FEAT_PMuV3 is implemented:**

Trap MRS reads and MRC reads of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MRS reads of [PMEVTYPER<n>_EL0](#) and [PMXEVTYPER_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MRC reads of [PMEVTYPER<n>](#) and [PMXEVTYPER](#).

PMEVTYPERn_EL0	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> • MRS reads at EL1 and EL0 using AArch64 of PMEVTYPER<n>_EL0 and PMXEVTYPER_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. • MRC reads at EL0 using AArch32 of PMEVTYPER<n> and PMXEVTYPER are trapped to EL2 and reported with EC syndrome value 0x03.

Regardless of the value of this field, for each value n:

- If event counter n is not implemented, the following accesses are UNDEFINED:
 - In AArch64 state, a read of [PMEVTYPER<n>_EL0](#), or, if n is not 31, a read of [PMXEVTYPER_EL0](#) when [PMSELR_EL0](#).SEL == n.
 - In AArch32 state, a read of [PMEVTYPER<n>](#), or, if n is not 31, a read of [PMXEVTYPER](#) when [PMSELR](#).SEL == n.
- If event counter n is implemented, n is greater-than-or-equal-to [MDCR_EL2](#).HPMN, and EL2 is implemented and enabled in the current Security state, the following generate a Trap exception to EL2 from EL0 or EL1:
 - In AArch64 state, a read of [PMEVTYPER<n>_EL0](#), or a read of [PMXEVTYPER_EL0](#) when [PMSELR_EL0](#).SEL == n, reported with EC syndrome value 0x18.
 - In AArch32 state, a read of [PMEVTYPER<n>](#), or a read of [PMXEVTYPER](#) when [PMSELR](#).SEL == n, reported with EC syndrome value 0x03.

See also [HDFGRTR_EL2.PMCCFILTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMEVCNTRn_EL0, bit [12]

When FEAT_PMuV3 is implemented:

Trap MRS reads and MRC reads of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MRS reads of [PMEVCNTR<n>_EL0](#) and [PMXEVCNTR_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MRC reads of [PMEVCNTR<n>](#) and [PMXEVCNTR](#).

PMEVCNTRn_EL0	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTE == 1, then, unless the read generates a higher priority exception:</p> <ul style="list-style-type: none"> • MRS reads at EL1 and EL0 using AArch64 of PMEVCNTR<n>_EL0 and PMXEVCNTR_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. • MRC reads at EL0 using AArch32 of PMEVCNTR<n> and PMXEVCNTR are trapped to EL2 and reported with EC syndrome value 0x03.

Regardless of the value of this field, for each value n:

- If event counter n is not implemented, the following accesses are UNDEFINED:
 - In AArch64 state, a read of [PMEVCNTR<n>_EL0](#), or a read of [PMXEVCNTR_EL0](#) when [PMSELR_EL0](#).SEL == n.
 - In AArch32 state, a read of [PMEVCNTR<n>](#), or a read of [PMXEVCNTR](#) when [PMSELR](#).SEL == n.
- If event counter n is implemented, n is greater-than-or-equal-to [MDCR_EL2](#).HPMN, and EL2 is implemented and enabled in the current Security state, the following generate a Trap exception to EL2 from EL0 or EL1:
 - In AArch64 state, a read of [PMEVCNTR<n>_EL0](#), or a read of [PMXEVCNTR_EL0](#) when [PMSELR_EL0](#).SEL == n, reported with EC syndrome value 0x18.
 - In AArch32 state, a read of [PMEVCNTR<n>](#), or a read of [PMXEVCNTR](#) when [PMSELR](#).SEL == n, reported with EC syndrome value 0x03.

PMEVCNTR<n>_EL0 is indirectly accessed when **PMCR_EL0.P** is set to 0b1.

Setting this field to 1 has no effect on accesses to **PMEVCNTR<n>_EL0** using **PMCR_EL0**.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

OSDLR_EL1, bit [11]

When FEAT_DoubleLock is implemented:

Trap MRS reads of [OSDLR_EL1](#) at EL1 using AArch64 to EL2.

OSDLR_EL1	Meaning
0b0	MRS reads of OSDLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of OSDLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

OSECCR_EL1, bit [10]

Trap MRS reads of [OSECCR_EL1](#) at EL1 using AArch64 to EL2.

OSECCR_EL1	Meaning
0b0	MRS reads of OSECCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of OSECCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

OSLSR_EL1, bit [9]

Trap MRS reads of [OSLSR_EL1](#) at EL1 using AArch64 to EL2.

OSLSR_EL1	Meaning
0b0	MRS reads of OSLSR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of OSLSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bit [8]

Reserved, RES0.

DBGPRCR_EL1, bit [7]

Trap MRS reads of [DBGPRCR_EL1](#) at EL1 using AArch64 to EL2.

DBGPRCR_EL1	Meaning
0b0	MRS reads of DBGPRCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGPRCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGAUTHSTATUS_EL1, bit [6]

Trap MRS reads of [DBGAUTHSTATUS_EL1](#) at EL1 using AArch64 to EL2.

DBGAUTHSTATUS_EL1	Meaning
0b0	MRS reads of DBGAUTHSTATUS_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGAUTHSTATUS_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGCLAIM, bit [5]

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [DBGCLAIMCLR_EL1](#).
- [DBGCLAIMSET_EL1](#).

DBGCLAIM	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

MDSCR_EL1, bit [4]

Trap MRS reads of [MDSCR_EL1](#) at EL1 using AArch64 to EL2.

MDSCR_EL1	Meaning
0b0	MRS reads of MDSCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MDSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGWVRn_EL1, bit [3]

Trap MRS reads of [DBGWVR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGWVRn_EL1	Meaning
0b0	MRS reads of DBGWVR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGWVR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If watchpoint n is not implemented, a read of [DBGWVR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGWCRn_EL1, bit [2]

Trap MRS reads of [DBGWCR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGWCRn_EL1	Meaning
0b0	MRS reads of DBGWCR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGWCR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If watchpoint n is not implemented, a read of [DBGWCR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGBVRn_EL1, bit [1]

Trap MRS reads of [DBGBVR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGBVRn_EL1	Meaning
0b0	MRS reads of DBGBVR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGBVR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If breakpoint n is not implemented, a read of [DBGBVR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGBCRn_EL1, bit [0]

Trap MRS reads of [DBGBCR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGBCRn_EL1	Meaning
0b0	MRS reads of DBGBCR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of DBGBCR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

If breakpoint n is not implemented, a read of [DBGBCR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing HDFGRTR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HDFGRTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1D0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HDFGRTR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HDFGRTR_EL2;

```

MSR HDFGRTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1D0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HDFGRTR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HDFGRTR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

HDFGWTR2_EL2, Hypervisor Debug Fine-Grained Write Trap Register 2

The HDFGWTR2_EL2 characteristics are:

Purpose

Provides controls for traps of MSR and MCR writes of debug, trace, PMU, and Statistical Profiling System registers.

Configuration

This register is present only when FEAT_FGT2 is implemented. Otherwise, direct accesses to HDFGWTR2_EL2 are UNDEFINED.

Attributes

HDFGWTR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46						
RES0				nPMZR_ELO				nTRCITECR_EL1				nPMSDSFR_EL1				RES0		nSPMSCR_EL1		nSPMACCESSR_EL1		nSPMCR_EL1	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14						

Bits [63:22]

Reserved, RES0.

nPMZR_ELO, bit [21]

When FEAT_PMUv3p9 is implemented:

Trap MSR writes of [PMZR_ELO](#) at EL1 and EL0 using AArch64 to EL2.

nPMZR_ELO	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes of PMZR_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMZR_ELO are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nTRCITECR_EL1, bit [20]**When FEAT_ITE is implemented:**

Trap MSR writes of [TRCITECR_EL1](#) at EL1 using AArch64 to EL2.

nTRCITECR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of TRCITECR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of TRCITECR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMDSFR_EL1, bit [19]**When FEAT_SPE_FDS is implemented:**

Trap MSR writes of [PMDSFR_EL1](#) at EL1 using AArch64 to EL2.

nPMDSFR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of PMDSFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMDSFR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [18:17]

Reserved, RES0.

nSPMSCR_EL1, bit [16]**When FEAT_SPMU is implemented:**

Trap MSR writes of [SPMSCR_EL1](#) at EL1 using AArch64 to EL2.

nSPMSCR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of SPMSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of SPMSCR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMACCESSR_EL1, bit [15]**When FEAT_SPMU is implemented:**

Trap MSR writes of [SPMACCESSR_EL1](#) at EL1 using AArch64 to EL2.

nSPMACCESSR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of SPMACCESSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of SPMACCESSR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMCR_EL0, bit [14]**When FEAT_SPMU is implemented:**

Trap MSR writes of [SPMCR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nSPMCR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes of SPMCR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of SPMCR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMOVS, bit [13]

When FEAT_SPMU is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMOVSLR_EL0](#).
- [SPMOVSSET_EL0](#).

nSPMOVS	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMINTEN, bit [12]

When FEAT_SPMU is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMINTENCLR_EL1](#).
- [SPMINTENSET_EL1](#).

nSPMINTEN	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMCNTEN, bit [11]

When FEAT_SPMU is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMCNTENCLR_EL0](#).
- [SPMCNTENSET_EL0](#).

nSPMCNTEN	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMSELR_EL0, bit [10]

When FEAT_SPMU is implemented:

Trap MSR writes of [SPMSELR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nSPMSELR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTE _{n2} == 1, then MSR writes of SPMSELR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of SPMSELR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMEVTYPEPn_EL0, bit [9]

When FEAT_SPMU is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2:

- [SPMEVTYPEP<n>_EL0](#).
- [SPMEVFILTR<n>_EL0](#).
- [SPMEVFILT2R<n>_EL0](#).

nSPMEVTYPEPn_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTE _{n2} == 1, then MSR writes at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

Regardless of the value of this bit, if event counter n is not implemented, a write of [SPMEVTYPEP<n>_EL0](#), [SPMEVFILTR<n>_EL0](#), or [SPMEVFILT2R<n>_EL0](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nSPMEVCNTRn_EL0, bit [8]**When FEAT_SPMU is implemented:**

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 and EL0 using AArch64 of any of the following AArch64 System registers to EL2: [SPMEVCNTR<n>_EL0](#).

nSPMEVCNTRn_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes at EL1 and EL0 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

Regardless of the value of this bit, if event counter n is not implemented, a write of [SPMEVCNTR<n>_EL0](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMSSCR_EL1, bit [7]**When FEAT_PMUv3_SS is implemented:**

Trap MSR writes of [PMSSCR_EL1](#) at EL1 using AArch64 to EL2.

nPMSSCR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes of PMSSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMSSCR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [6]

Reserved, RES0.

nMDSELR_EL1, bit [5]**When FEAT_Debugv8p9 is implemented:**

Trap MSR writes of [MDSELR_EL1](#) at EL1 using AArch64 to EL2.

nMDSELR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of MDSELR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of MDSELR_EL1 are not trapped by this mechanism.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and [MDSELR_EL1](#) is implemented as RAZ/WI.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMUACR_EL1, bit [4]**When FEAT_PMUv3p9 is implemented:**

Trap MSR writes of [PMUACR_EL1](#) at EL1 using AArch64 to EL2.

nPMUACR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of PMUACR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMUACR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMICFILTR_EL0, bit [3]**When FEAT_PMUv3_ICNTR is implemented:**

Trap MSR writes of [PMICFILTR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nPMICFILTR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes of PMICFILTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMICFILTR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMICNTR_EL0, bit [2]

When FEAT_PMuV3_ICNTR is implemented:

Trap MSR writes of [PMICNTR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nPMICNTR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MSR writes of PMICNTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMICNTR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMIAR_EL1, bit [1]

When FEAT_SEBEP is implemented:

Trap MSR writes of [PMIAR_EL1](#) at EL1 using AArch64 to EL2.

nPMIAR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of PMIAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMIAR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPMECR_EL1, bit [0]

When FEAT_EBEP is implemented:

Trap MSR writes of [PMECR_EL1](#) at EL1 using AArch64 to EL2.

nPMECR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of PMECR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMECR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing HDFGWTR2_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HDFGWTR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1B0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HDFGWTR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HDFGWTR2_EL2;

```

MSR HDFGWTR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1B0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HDFGWTR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HDFGWTR2_EL2 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HDFGWTR_EL2, Hypervisor Debug Fine-Grained Write Trap Register

The HDFGWTR_EL2 characteristics are:

Purpose

Provides controls for traps of MSR and MCR writes of debug, trace, PMU, and Statistical Profiling System registers.

Configuration

This register is present only when FEAT_FGT is implemented. Otherwise, direct accesses to HDFGWTR_EL2 are UNDEFINED.

Attributes

HDFGWTR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	
RES0	nPMSNEVFR_EL1	nBRBDATA	nBRBCTL	RES0		PMUSERENR_ELO	TRBTRG_EL1	TF
PMSIRR_EL1	RES0	PMSICR_EL1	PMSFCR_EL1	PMSEVFR_EL1	PMSCR_EL1	PMBSR_EL1	PMBPTR_EL1	PMB
31	30	29	28	27	26	25	24	

Bit [63]

Reserved, RES0.

nPMSNEVFR_EL1, bit [62] When FEAT_SPEv1p2 is implemented:

Trap MSR writes of [PMSNEVFR_EL1](#) at EL1 using AArch64 to EL2.

nPMSNEVFR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MSR writes of PMSNEVFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PMSNEVFR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBDATA, bit [61]**When FEAT_BRBE is implemented:**

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [BRBINFINJ_EL1](#).
- [BRBSRCINJ_EL1](#).
- [BRBTGTINJ_EL1](#).
- [BRBTS_EL1](#).

nBRBDATA	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBCTL, bit [60]**When FEAT_BRBE is implemented:**

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [BRBCR_EL1](#).
- [BRBFCR_EL1](#).

nBRBCTL	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of the System registers listed above are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [59:58]

Reserved, RES0.

PMUSERENR_EL0, bit [57]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMUSERENR_EL0](#) at EL1 using AArch64 to EL2.

PMUSERENR_EL0	Meaning
0b0	MSR writes of PMUSERENR_EL0 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMUSERENR_EL0 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBTRG_EL1, bit [56]**When FEAT_TRBE is implemented:**

Trap MSR writes of [TRBTRG_EL1](#) at EL1 using AArch64 to EL2.

TRBTRG_EL1	Meaning
0b0	MSR writes of TRBTRG_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBTRG_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBSR_EL1, bit [55]**When FEAT_TRBE is implemented:**

Trap MSR writes of [TRBSR_EL1](#) at EL1 using AArch64 to EL2.

TRBSR_EL1	Meaning
0b0	MSR writes of TRBSR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBPTR_EL1, bit [54]

When FEAT_TRBE is implemented:

Trap MSR writes of [TRBPTR_EL1](#) at EL1 using AArch64 to EL2.

TRBPTR_EL1	Meaning
0b0	MSR writes of TRBPTR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBPTR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBMAR_EL1, bit [53]

When FEAT_TRBE is implemented:

Trap MSR writes of [TRBMAR_EL1](#) at EL1 using AArch64 to EL2.

TRBMAR_EL1	Meaning
0b0	MSR writes of TRBMAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBMAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRBLIMITR_EL1, bit [52]**When FEAT_TRBE is implemented:**

Trap MSR writes of [TRBLIMITR_EL1](#) at EL1 using AArch64 to EL2.

TRBLIMITR_EL1	Meaning
0b0	MSR writes of TRBLIMITR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBLIMITR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [51]

Reserved, RES0.

TRBBASER_EL1, bit [50]**When FEAT_TRBE is implemented:**

Trap MSR writes of [TRBBASER_EL1](#) at EL1 using AArch64 to EL2.

TRBBASER_EL1	Meaning
0b0	MSR writes of TRBBASER_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRBBASER_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRFCR_EL1, bit [49]**When FEAT_TRF is implemented:**

Trap MSR writes of [TRFCR_EL1](#) at EL1 using AArch64 to EL2.

TRFCR_EL1	Meaning
0b0	MSR writes of TRFCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRFCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCVICTLR, bit [48]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCVICTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCVICTLR at EL1 using AArch64 to EL2.

TRCVICTLR	Meaning
0b0	MSR writes of TRCVICTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCVICTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [47]

Reserved, RES0.

TRCSSCSRn, bit [46]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCSSCSR<n> are implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCSSCSR<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCSSCSR<n> at EL1 using AArch64 to EL2.

TRCSSCSRn	Meaning
0b0	MSR writes of TRCSSCSR<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCSSCSR<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If Single-shot Comparator n is not implemented, a write of [TRCSSCSR<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCSEQSTR, bit [45]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCSEQSTR is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCSEQSTR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCSEQSTR at EL1 using AArch64 to EL2.

TRCSEQSTR	Meaning
0b0	MSR writes of TRCSEQSTR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCSEQSTR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCPRGCTLR, bit [44]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCPRGCTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCPRGCTLR at EL1 using AArch64 to EL2.

TRCPRGCTLR	Meaning
0b0	MSR writes of TRCPRGCTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCPRGCTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [43]

Reserved, RES0.

TRCOSLAR, bit [42]

When System register access to the trace unit registers is implemented and FEAT_ETMv4 is implemented:

In an Armv8 implementation, trap MSR writes of ETM TRCOSLAR at EL1 using AArch64 to EL2.

TRCOSLAR	Meaning
0b0	MSR writes of TRCOSLAR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCOSLAR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCIMSPECn, bit [41]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCIMSPEC<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCIMSPEC<n> at EL1 using AArch64 to EL2.

TRCIMSPECn	Meaning
0b0	MSR writes of TRCIMSPEC<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCIMSPEC<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

TRCIMSPEC<1-7> are optional. If [TRCIMSPEC<n>](#) is not implemented, a write of [TRCIMSPEC<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [40:38]

Reserved, RES0.

TRCCNTVRn, bit [37]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented, TRCCNTVR<n> are implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCCNTVR<n>](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCCNTVR<n> at EL1 using AArch64 to EL2.

TRCCNTVRn	Meaning
0b0	MSR writes of TRCCNTVR<n> are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCCNTVR<n> at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If Counter n is not implemented, a write of [TRCCNTVR<n>](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCCLAIM, bit [36]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCCLAIMCLR](#) and [TRCCLAIMSET](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCCLAIMCLR and ETM TRCCLAIMSET at EL1 using AArch64 to EL2.

TRCCLAIM	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCAUXCTLR, bit [35]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of [TRCAUXCTLR](#) at EL1 using AArch64 to EL2.

In an Armv8 implementation, trap MSR writes of ETM TRCAUXCTLR at EL1 using AArch64 to EL2.

TRCAUXCTLR	Meaning
0b0	MSR writes of TRCAUXCTLR are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TRCAUXCTLR at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [34]

Reserved, RES0.

TRC, bit [33]

When FEAT_ETE is implemented or (FEAT_ETMv4 is implemented and System register access to the trace unit registers is implemented):

In an Armv9 implementation, trap MSR writes of the following registers at EL1 using AArch64 to EL2:

- [TRCACATR<n>](#).
- [TRCACVR<n>](#).
- [TRCBBCTLR](#).
- [TRCCCCTLR](#).
- [TRCCIDCTLRO](#).
- [TRCCIDCTLRI](#).

- [TRCCIDCVR<n>](#).
- [TRCCNTCTLR<n>](#).
- [TRCCNTRLDVR<n>](#).
- [TRCCONFIGR](#).
- [TRCEVENTCTL0R](#).
- [TRCEVENTCTL1R](#).
- [TRCEXTINSELR<n>](#).
- [TRCQCTLR](#).
- [TRCRSCTLR<n>](#).
- [TRCRSR](#).
- [TRCSEQEVR<n>](#).
- [TRCSEQRSTEVR](#).
- [TRCSSCCR<n>](#).
- [TRCSSPCICR<n>](#).
- [TRCSTALLCTLR](#).
- [TRCSYNCPR](#).
- [TRCTRACEIDR](#).
- [TRCTSCTLR](#).
- [TRCVIIECTLR](#).
- [TRCVIPCSSCTLR](#).
- [TRCVISSCTLR](#).
- [TRCVMIDCCTL0R](#).
- [TRCVMIDCCTL1R](#).
- [TRCVMIDCVR<n>](#).

In an Armv8 implementation, trap MSR writes of the following registers at EL1 using AArch64 to EL2:

- ETM TRCACATR<n>.
- ETM TRCACVR<n>.
- ETM TRCBBCTLR.
- ETM TRCCCCTLR.
- ETM TRCCIDCCTL0R.
- ETM TRCCIDCCTL1R.
- ETM TRCCIDCVR<n>.
- ETM TRCCNTCTLR<n>.
- ETM TRCCNTRLDVR<n>.
- ETM TRCCONFIGR.
- ETM TRCEVENTCTL0R.
- ETM TRCEVENTCTL1R.
- ETM TRCEXTINSELR.
- ETM TRCQCTLR.
- ETM TRCRSCTLR<n>.
- ETM TRCSEQEVR<n>.
- ETM TRCSEQRSTEVR.
- ETM TRCSSCCR<n>.
- ETM TRCSSPCICR<n>.
- ETM TRCSTALLCTLR.
- ETM TRCSYNCPR.
- ETM TRCTRACEIDR.
- ETM TRCTSCTLR.
- ETM TRCVIIECTLR.
- ETM TRCVIPCSSCTLR.
- ETM TRCVISSCTLR.
- ETM TRCVMIDCCTL0R.
- ETM TRCVMIDCCTL1R.
- ETM TRCVMIDCVR<n>.

TRC	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

A write of an unimplemented register is UNDEFINED.

[TRCEXTINSELR<n>](#) and [TRCRSR](#) are only implemented if FEAT_ETE is implemented.

TRCEXTINSEL is only implemented if FEAT_ETE is not implemented and FEAT_ETMv4 is implemented.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSLATFR_EL1, bit [32]

When FEAT_SPE is implemented:

Trap MSR writes of [PMSLATFR_EL1](#) at EL1 using AArch64 to EL2.

PMSLATFR_EL1	Meaning
0b0	MSR writes of PMSLATFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSLATFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSIRR_EL1, bit [31]

When FEAT_SPE is implemented:

Trap MSR writes of [PMSIRR_EL1](#) at EL1 using AArch64 to EL2.

PMSIRR_EL1	Meaning
0b0	MSR writes of PMSIRR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSIRR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [30]

Reserved, RES0.

PMSICR_EL1, bit [29]**When FEAT_SPE is implemented:**

Trap MSR writes of [PMSICR_EL1](#) at EL1 using AArch64 to EL2.

PMSICR_EL1	Meaning
0b0	MSR writes of PMSICR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSICR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSEVFR_EL1, bit [28]**When FEAT_SPE is implemented:**

Trap MSR writes of [PMSEVFR_EL1](#) at EL1 using AArch64 to EL2.

PMSEVFR_EL1	Meaning
0b0	MSR writes of PMSEVFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSEVFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSEVFR_EL1, bit [27]**When FEAT_SPE is implemented:**

Trap MSR writes of [PMSEVFR_EL1](#) at EL1 using AArch64 to EL2.

PMSEVFR_EL1	Meaning
0b0	MSR writes of PMSEVFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSEVFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSCR_EL1, bit [26]

When FEAT_SPE is implemented:

Trap MSR writes of [PMSCR_EL1](#) at EL1 using AArch64 to EL2.

PMSCR_EL1	Meaning
0b0	MSR writes of PMSCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBSR_EL1, bit [25]

When FEAT_SPE is implemented:

Trap MSR writes of [PMBSR_EL1](#) at EL1 using AArch64 to EL2.

PMBSR_EL1	Meaning
0b0	MSR writes of PMBSR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMBSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBPTR_EL1, bit [24]**When FEAT_SPE is implemented:**

Trap MSR writes of [PMBPTR_EL1](#) at EL1 using AArch64 to EL2.

PMBPTR_EL1	Meaning
0b0	MSR writes of PMBPTR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMBPTR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMBLIMITR_EL1, bit [23]**When FEAT_SPE is implemented:**

Trap MSR writes of [PMBLIMITR_EL1](#) at EL1 using AArch64 to EL2.

PMBLIMITR_EL1	Meaning
0b0	MSR writes of PMBLIMITR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PMBLIMITR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [22]

Reserved, RES0.

PMCR_ELO, bit [21]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMCR_ELO](#) at EL1 and EL0 using AArch64 and MCR writes of [PMCR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCR_ELO	Meaning
0b0	MSR writes of PMCR_ELO at EL1 and EL0 using AArch64 and MCR writes of PMCR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of PMCR_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes of PMCR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSWINC_ELO, bit [20]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMSWINC_ELO](#) at EL1 and EL0 using AArch64 and MCR writes of [PMSWINC](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMSWINC_ELO	Meaning
0b0	MSR writes of PMSWINC_ELO at EL1 and EL0 using AArch64 and MCR writes of PMSWINC at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of PMSWINC_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes of PMSWINC at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMSELR_EL0, bit [19]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMSELR_EL0](#) at EL1 and EL0 using AArch64 and MCR writes of [PMSELR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMSELR_EL0	Meaning
0b0	MSR writes of PMSELR_EL0 at EL1 and EL0 using AArch64 and MCR writes of PMSELR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of PMSELR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes of PMSELR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMOVS, bit [18]**When FEAT_PMUv3 is implemented:**

Trap MSR writes and MCR writes of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MSR writes of [PMOVSLR_EL0](#) and [PMOVSSET_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MCR writes of [PMOVS](#) and [PMOVSSET](#).

PMOVS	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes at EL1 and EL0 using AArch64 of PMOVSLR_EL0 and PMOVSSET_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes at EL0 using AArch32 of PMOVS and PMOVSSET are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMINTEN, bit [17]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [PMINTENCLR_EL1](#).
- [PMINTENSET_EL1](#).

PMINTEN	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCNTEN, bit [16]**When FEAT_PMUv3 is implemented:**

Trap MSR writes and MCR writes of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MSR writes of [PMCNTENCLR_EL0](#) and [PMCNTENSET_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MCR writes of [PMCNTENCLR](#) and [PMCNTENSET](#).

PMCNTEN	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> • MSR writes at EL1 and EL0 using AArch64 of PMCNTENCLR_EL0 and PMCNTENSET_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. • MCR writes at EL0 using AArch32 of PMCNTENCLR and PMCNTENSET are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCCNTR_EL0, bit [15]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMCCNTR_EL0](#) at EL1 and EL0 using AArch64 and MCR and MCRR writes of [PMCCNTR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCCNTR_EL0	Meaning
0b0	MSR writes of PMCCNTR_EL0 at EL1 and EL0 using AArch64 and MCR and MCRR writes of PMCCNTR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of PMCCNTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR and MCRR writes of PMCCNTR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03 (for MCR) or 0x04 (for MCRR).

[PMCCNTR_EL0](#) is indirectly accessed when [PMCR_EL0.C](#) is set to 0b1.

Setting this field to 1 has no effect on accesses to [PMCCNTR_EL0](#) using [PMCR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMCCFILTR_EL0, bit [14]**When FEAT_PMUv3 is implemented:**

Trap MSR writes of [PMCCFILTR_EL0](#) at EL1 and EL0 using AArch64 and MCR writes of [PMCCFILTR](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

PMCCFILTR_EL0	Meaning
0b0	MSR writes of PMCCFILTR_EL0 at EL1 and EL0 using AArch64 and MCR writes of PMCCFILTR at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of PMCCFILTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes of PMCCFILTR at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

[PMCCFILTR_EL0](#) can also be accessed in AArch64 state using [PMXEVTYPER_EL0](#) when [PMSELR_EL0](#).SEL == 31, and [PMCCFILTR](#) can also be accessed in AArch32 state using [PMXEVTYPER](#) when [PMSELR](#).SEL == 31.

Setting this field to 1 has no effect on accesses to [PMXEVTYPER_EL0](#) and [PMXEVTYPER](#), regardless of the value of [PMSELR_EL0](#).SEL or [PMSELR](#).SEL.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMEVTYPEPn_EL0, bit [13]

When FEAT_PMUv3 is implemented:

Trap MSR writes and MCR writes of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MSR writes of [PMEVTYPEP<n>_EL0](#) and [PMXEVTYPEn_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MCR writes of [PMEVTYPEP<n>](#) and [PMXEVTYPEn](#).

PMEVTYPEPn_EL0	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEN == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes at EL1 and EL0 using AArch64 of PMEVTYPEP<n>_EL0 and PMXEVTYPEn_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes at EL0 using AArch32 of PMEVTYPEP<n> and PMXEVTYPEn are trapped to EL2 and reported with EC syndrome value 0x03.

Regardless of the value of this field, for each value n:

- If event counter n is not implemented, the following accesses are UNDEFINED:
 - In AArch64 state, a write of [PMEVTYPEP<n>_EL0](#), or, if n is not 31, a write of [PMXEVTYPEn_EL0](#) when [PMSELR_EL0](#).SEL == n.
 - In AArch32 state, a write of [PMEVTYPEP<n>](#), or, if n is not 31, a write of [PMXEVTYPEn](#) when [PMSELR](#).SEL == n.
- If event counter n is implemented, n is greater-than-or-equal-to [MDCR_EL2](#).HPMN, and EL2 is implemented and enabled in the current Security state, the following generate a Trap exception to EL2 from EL0 or EL1:
 - In AArch64 state, a write of [PMEVTYPEP<n>_EL0](#), or a write of [PMXEVTYPEn_EL0](#) when [PMSELR_EL0](#).SEL == n, reported with EC syndrome value 0x18.
 - In AArch32 state, a write of [PMEVTYPEP<n>](#), or a write of [PMXEVTYPEn](#) when [PMSELR](#).SEL == n, reported with EC syndrome value 0x03.

See also HDFGWTR_EL2.PMCCFILTR_EL0.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

PMEVCNTRn_EL0, bit [12]**When FEAT_PMUv3 is implemented:**

Trap MSR writes and MCR writes of multiple System registers.

Enables a trap to EL2 the following operations:

- At EL1 and EL0 using AArch64: MSR writes of [PMEVCNTR<n>_EL0](#) and [PMXEVCNTR_EL0](#).
- At EL0 using AArch32 when EL1 is using AArch64: MCR writes of [PMEVCNTR<n>](#) and [PMXEVCNTR](#).

PMEVCNTRn_EL0	Meaning
0b0	The operations listed above are not trapped by this mechanism.
0b1	<p>If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the write generates a higher priority exception:</p> <ul style="list-style-type: none"> • MSR writes at EL1 and EL0 using AArch64 of PMEVCNTR<n>_EL0 and PMXEVCNTR_EL0 are trapped to EL2 and reported with EC syndrome value 0x18. • MCR writes at EL0 using AArch32 of PMEVCNTR<n> and PMXEVCNTR are trapped to EL2 and reported with EC syndrome value 0x03.

Regardless of the value of this field, for each value n:

- If event counter n is not implemented, the following accesses are UNDEFINED:
 - In AArch64 state, a write of [PMEVCNTR<n>_EL0](#), or a write of [PMXEVCNTR_EL0](#) when [PMSELR_EL0](#).SEL == n.
 - In AArch32 state, a write of [PMEVCNTR<n>](#), or a write of [PMXEVCNTR](#) when [PMSELR](#).SEL == n.
- If event counter n is implemented, n is greater-than-or-equal-to [MDCR_EL2](#).HPMN, and EL2 is implemented and enabled in the current Security state, the following generate a Trap exception to EL2 from EL0 or EL1:
 - In AArch64 state, a write of [PMEVCNTR<n>_EL0](#), or a write of [PMXEVCNTR_EL0](#) when [PMSELR_EL0](#).SEL == n, reported with EC syndrome value 0x18.
 - In AArch32 state, a write of [PMEVCNTR<n>](#), or a write of [PMXEVCNTR](#) when [PMSELR](#).SEL == n, reported with EC syndrome value 0x03.

[PMEVCNTR<n>_EL0](#) is indirectly accessed when [PMCR_EL0](#).P is set to 0b1.

Setting this field to 1 has no effect on accesses to [PMEVCNTR<n>_EL0](#) using [PMCR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

OSDLR_EL1, bit [11]**When FEAT_DoubleLock is implemented:**

Trap MSR writes of [OSDLR_EL1](#) at EL1 using AArch64 to EL2.

OSDLR_EL1	Meaning
0b0	MSR writes of OSDLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of OSDLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

OSECCR_EL1, bit [10]

Trap MSR writes of [OSECCR_EL1](#) at EL1 using AArch64 to EL2.

OSECCR_EL1	Meaning
0b0	MSR writes of OSECCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of OSECCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bit [9]

Reserved, RES0.

OSLAR_EL1, bit [8]

Trap MSR writes of [OSLAR_EL1](#) at EL1 using AArch64 to EL2.

OSLAR_EL1	Meaning
0b0	MSR writes of OSLAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of OSLAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGPRCR_EL1, bit [7]

Trap MSR writes of [DBGPRCR_EL1](#) at EL1 using AArch64 to EL2.

DBGPRCR_EL1	Meaning
0b0	MSR writes of DBGPRCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of DBGPRCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bit [6]

Reserved, RES0.

DBGCLAIM, bit [5]

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [DBGCLAIMCLR_EL1](#).
- [DBGCLAIMSET_EL1](#).

DBGCLAIM	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

MDSCR_EL1, bit [4]

Trap MSR writes of [MDSCR_EL1](#) at EL1 using AArch64 to EL2.

MDSCR_EL1	Meaning
0b0	MSR writes of MDSCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of MDSCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGWVRn_EL1, bit [3]

Trap MSR writes of [DBGWVR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGWVRn_EL1	Meaning
0b0	MSR writes of DBGWVR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of DBGWVR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If watchpoint n is not implemented, a write of [DBGWVR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGWCRn_EL1, bit [2]

Trap MSR writes of [DBGWCR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGWCRn_EL1	Meaning
0b0	MSR writes of DBGWCR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of DBGWCR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If watchpoint n is not implemented, a write of [DBGWCR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGBVRn_EL1, bit [1]

Trap MSR writes of [DBGBVR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGBVRn_EL1	Meaning
0b0	MSR writes of DBGBVR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of DBGBVR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If breakpoint n is not implemented, a write of [DBGBVR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DBGBCRn_EL1, bit [0]

Trap MSR writes of [DBGBCR<n>_EL1](#) at EL1 using AArch64 to EL2.

DBGBCRn_EL1	Meaning
0b0	MSR writes of DBGBCR<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of DBGBCR<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

If breakpoint n is not implemented, a write of [DBGBCR<n>_EL1](#) is UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing HDFGWTR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HDFGWTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1D8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HDFGWTR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HDFGWTR_EL2;

```

MRS HDFGWTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1D8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HDFGWTR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HDFGWTR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

HFGITR2_EL2, Hypervisor Fine-Grained Instruction Trap Register 2

The HFGITR2_EL2 characteristics are:

Purpose

Provides instruction trap controls.

Configuration

This register is present only when FEAT_FGT2 is implemented. Otherwise, direct accesses to HFGITR2_EL2 are UNDEFINED.

Attributes

HFGITR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Reserved, RES0.

Accessing HFGITR2_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGITR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x310];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HFGITR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGITR2_EL2;
```

MSR HFGITR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x310] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HFGITR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGITR2_EL2 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HFGITR_EL2, Hypervisor Fine-Grained Instruction Trap Register

The HFGITR_EL2 characteristics are:

Purpose

Provides instruction trap controls.

Configuration

This register is present only when FEAT_FGT is implemented. Otherwise, direct accesses to HFGITR_EL2 are UNDEFINED.

Attributes

HFGITR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56
	RES0		COSPRCTXnBRBIAL		RES0nBRBIN		nBRBIAL
TLBIVAAE1IS	TLBIASIDE1IS	TLBIVAE1IS	TLBIVMALE1IS	TLBIRVALE1OS	TLBIRVAE1OS	TLBIRVAE1OS	TLBIRVA
31	30	29	28	27	26	25	24

Bits [63:6157]

Reserved, RES0.

COSPRCTX, bit [60]

When FEAT_SPECRES2 is implemented:

Trap execution of [COSP_RCTX](#) at EL1 and EL0 using AArch64 and execution of [COSPRCTX](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

COSPRCTX	Meaning
0b0	Execution of COSP_RCTX at EL1 and EL0 using AArch64 and execution of COSPRCTX at EL0 using AArch32 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTE == 1, then, unless the instruction generates a higher priority exception: <ul style="list-style-type: none"> Execution of COSP_RCTX at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18. Execution of COSPRCTX at EL0 using AArch32 is trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [59:57]

Reserved, RES0.

nBRBIALL, bit [56]**When FEAT_BRBE is implemented:**Trap execution of [BRB IALL](#) at EL1 using AArch64 to EL2.

nBRBIALL	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of BRB IALL at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.
0b1	Execution of BRB IALL is not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nBRBINJ, bit [55]**When FEAT_BRBE is implemented:**Trap execution of [BRB INJ](#) at EL1 using AArch64 to EL2.

nBRBINJ	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of BRB INJ at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.
0b1	Execution of BRB INJ is not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

DCCVAC, bit [54]

Trap execution of multiple instructions. Enables a trap on execution at EL1 and EL0 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CVAC](#).

- [DC_CGVAC](#), if FEAT_MTE is implemented.
- [DC_CGDVAC](#), if FEAT_MTE is implemented.

If the Point of Coherence is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCCVAC	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 and EL0 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

SVC_EL1, bit [53]

Trap execution of SVC at EL1 using AArch64 to EL2.

SVC_EL1	Meaning
0b0	Execution of SVC is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of SVC at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x15, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

SVC_EL0, bit [52]

Trap execution of SVC at EL0 using AArch64 and execution of SVC at EL0 using AArch32 when EL1 is using AArch64 to EL2.

SVC_EL0	Meaning
0b0	Execution of SVC at EL0 using AArch64 and execution of SVC at EL0 using AArch32 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then, unless the instruction generates a higher priority exception: <ul style="list-style-type: none"> • Execution of SVC at EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x15. • Execution of SVC at EL0 using AArch32 is trapped to EL2 and reported with EC syndrome value 0x11.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ERET, bit [51]

Trap execution of multiple instructions. Enables a trap on execution at EL1 using AArch64 of any of the following AArch64 instructions to EL2:

- ERET.

- ERETAA, if FEAT_PAuth is implemented.
- ERETAB, if FEAT_PAuth is implemented.

ERET	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution at EL1 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x1A, unless the instruction generates a higher priority exception.

If EL2 is implemented and enabled in the current Security state, [HCR_EL2.API](#) == 0, and this field enables a fine-grained trap on the instruction, then execution at EL1 using AArch64 of ERETAA or ERETAB instructions is trapped to EL2 and reported with EC syndrome value 0x1A with its associated ISS field, as the fine-grained trap has higher priority than the trap enabled by [HCR_EL2.API](#) == 0.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CPPRCTX, bit [50]

When FEAT_SPECRES is implemented:

Trap execution of [CPP RCTX](#) at EL1 and EL0 using AArch64 and execution of [CPPRCTX](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

CPPRCTX	Meaning
0b0	Execution of CPP RCTX at EL1 and EL0 using AArch64 and execution of CPPRCTX at EL0 using AArch32 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the instruction generates a higher priority exception: <ul style="list-style-type: none"> • Execution of CPP RCTX at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18. • Execution of CPPRCTX at EL0 using AArch32 is trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

DVPRCTX, bit [49]

When FEAT_SPECRES is implemented:

Trap execution of [DVP RCTX](#) at EL1 and EL0 using AArch64 and execution of [DVPRCTX](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

DVPRCTX	Meaning
0b0	Execution of DVP RCTX at EL1 and EL0 using AArch64 and execution of DVPRCTX at EL0 using AArch32 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTE == 1, then, unless the instruction generates a higher priority exception: <ul style="list-style-type: none"> Execution of DVP RCTX at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18. Execution of DVPRCTX at EL0 using AArch32 is trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

CFPRCTX, bit [48]

When FEAT_SPECRES is implemented:

Trap execution of [CFP RCTX](#) at EL1 and EL0 using AArch64 and execution of [CFPRCTX](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

CFPRCTX	Meaning
0b0	Execution of CFP RCTX at EL1 and EL0 using AArch64 and execution of CFPRCTX at EL0 using AArch32 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3 .FGTE == 1, then, unless the instruction generates a higher priority exception: <ul style="list-style-type: none"> Execution of CFP RCTX at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18. Execution of CFPRCTX at EL0 using AArch32 is trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVAALE1, bit [47]

Trap execution of [TLBI VAALE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2](#).FGTnXS == 0, this field also traps execution of TLBI VAALE1NXS.

TLBIVAAE1	Meaning
0b0	Execution of TLBI VAAE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAAE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVALE1, bit [46]

Trap execution of [TLBI VALE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VALE1NXS.

TLBIVALE1	Meaning
0b0	Execution of TLBI VALE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VALE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVAAE1, bit [45]

Trap execution of [TLBI VAAE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAAE1NXS.

TLBIVAAE1	Meaning
0b0	Execution of TLBI VAAE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAAE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIASIDE1, bit [44]

Trap execution of [TLBI ASIDE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI ASIDE1NXS.

TLBIASIDE1	Meaning
0b0	Execution of TLBI ASIDE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI ASIDE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVAE1, bit [43]

Trap execution of [TLBI VAE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAE1NXS.

TLBIVAE1	Meaning
0b0	Execution of TLBI VAE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVMALE1, bit [42]

Trap execution of [TLBI VMALLE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VMALLE1NXS.

TLBIVMALE1	Meaning
0b0	Execution of TLBI VMALLE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VMALLE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIRVALE1, bit [41]

When FEAT_TLBIRANGE is implemented:

Trap execution of [TLBI RVAALE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVAALE1NXS.

TLBIRVALE1	Meaning
0b0	Execution of TLBI RVALE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVALE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVALE1, bit [40]

When FEAT_TLBIRANGE is implemented:

Trap execution of [TLBI RVALE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVALE1NXS.

TLBIRVALE1	Meaning
0b0	Execution of TLBI RVALE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVALE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAE1, bit [39]

When FEAT_TLBIRANGE is implemented:

Trap execution of [TLBI RVAE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVAE1NXS.

TLBIRVAE1	Meaning
0b0	Execution of TLBI RVAE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVAE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAE1, bit [38]**When FEAT_TLBIRANGE is implemented:**

Trap execution of [TLBI RVAE1](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVAE1NXS.

TLBIRVAE1	Meaning
0b0	Execution of TLBI RVAE1 is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVAE1 at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAALE1IS, bit [37]**When FEAT_TLBIRANGE is implemented:**

Trap execution of [TLBI RVAALE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVAALE1ISNXS.

TLBIRVAALE1IS	Meaning
0b0	Execution of TLBI RVAALE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVAALE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVALE1IS, bit [36]**When FEAT_TLBIRANGE is implemented:**

Trap execution of [TLBI RVALE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2](#).FGTnXS == 0, this field also traps execution of TLBI RVALE1ISNXS.

TLBIRVALE1IS	Meaning
0b0	Execution of TLBI RVALE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of TLBI RVALE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAAE1IS, bit [35]**When FEAT_TLBIRANGE is implemented:**

Trap execution of [TLBI RVAAE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2](#).FGTnXS == 0, this field also traps execution of TLBI RVAAE1ISNXS.

TLBIRVAAE1IS	Meaning
0b0	Execution of TLBI RVAAE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of TLBI RVAAE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAE1IS, bit [34]**When FEAT_TLBIRANGE is implemented:**

Trap execution of [TLBI RVAE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2](#).FGTnXS == 0, this field also traps execution of TLBI RVAE1ISNXS.

TLBIRVAE1IS	Meaning
0b0	Execution of TLBI RVAE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVAE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVAALE1IS, bit [33]

Trap execution of [TLBI VAALE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAALE1ISNXS.

TLBIVAALE1IS	Meaning
0b0	Execution of TLBI VAALE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAALE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVALE1IS, bit [32]

Trap execution of [TLBI VALE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VALE1ISNXS.

TLBIVALE1IS	Meaning
0b0	Execution of TLBI VALE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VALE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVAAE1IS, bit [31]

Trap execution of [TLBI VAAE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAAE1ISNXS.

TLBIVAAE1IS	Meaning
0b0	Execution of TLBI VAAE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAAE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIASIDE1IS, bit [30]

Trap execution of [TLBI ASIDE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI ASIDE1ISNXS.

TLBIASIDE1IS	Meaning
0b0	Execution of TLBI ASIDE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI ASIDE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVAE1IS, bit [29]

Trap execution of [TLBI VAE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAE1ISNXS.

TLBIVAE1IS	Meaning
0b0	Execution of TLBI VAE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIVMALE1IS, bit [28]

Trap execution of [TLBI VMALLE1IS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VMALLE1ISNXS.

TLBIVMALE1IS	Meaning
0b0	Execution of TLBI VMALE1IS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VMALE1IS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TLBIRVALE1OS, bit [27]

When FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented:

Trap execution of [TLBI RVALE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVALE1OSNXS.

TLBIRVALE1OS	Meaning
0b0	Execution of TLBI RVALE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVALE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVALE1OS, bit [26]

When FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented:

Trap execution of [TLBI RVALE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RVALE1OSNXS.

TLBIRVALE1OS	Meaning
0b0	Execution of TLBI RVALE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RVALE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAAE1OS, bit [25]**When FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented:**

Trap execution of [TLBI RAAE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RAAE1OSNXS.

TLBIRVAAE1OS	Meaning
0b0	Execution of TLBI RAAE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RAAE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIRVAE1OS, bit [24]**When FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented:**

Trap execution of [TLBI RAE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI RAE1OSNXS.

TLBIRVAE1OS	Meaning
0b0	Execution of TLBI RAE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI RAE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVAAE1OS, bit [23]**When FEAT_TLBIOS is implemented:**

Trap execution of [TLBI VAAE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAALE1OSNXS.

TLBIVAALE1OS	Meaning
0b0	Execution of TLBI VAALE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAALE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVALE1OS, bit [22]

When FEAT_TLBIOS is implemented:

Trap execution of [TLBI VALE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VALE1OSNXS.

TLBIVALE1OS	Meaning
0b0	Execution of TLBI VALE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VALE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVAAE1OS, bit [21]

When FEAT_TLBIOS is implemented:

Trap execution of [TLBI VAAE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAAE1OSNXS.

TLBIVAAE1OS	Meaning
0b0	Execution of TLBI VAAE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAAE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIASIDE1OS, bit [20]

When FEAT_TLBIOS is implemented:

Trap execution of [TLBI ASIDE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI ASIDE1OSNXS.

TLBIASIDE1OS	Meaning
0b0	Execution of TLBI ASIDE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI ASIDE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBIVAE1OS, bit [19]

When FEAT_TLBIOS is implemented:

Trap execution of [TLBI VAE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2.FGTnXS](#) == 0, this field also traps execution of TLBI VAE1OSNXS.

TLBIVAE1OS	Meaning
0b0	Execution of TLBI VAE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of TLBI VAE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

TLBVMALLE1OS, bit [18]**When FEAT_TLBIOS is implemented:**

Trap execution of [TLBI VMALLE1OS](#) at EL1 using AArch64 to EL2.

If FEAT_XS is implemented and [HCRX_EL2](#).FGTnXS == 0, this field also traps execution of TLBI VMALLE1OSNXS.

TLBVMALLE1OS	Meaning
0b0	Execution of TLBI VMALLE1OS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of TLBI VMALLE1OS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ATS1E1WP, bit [17]**When FEAT_PAN2 is implemented:**

Trap execution of [AT S1E1WP](#) at EL1 using AArch64 to EL2.

ATS1E1WP	Meaning
0b0	Execution of AT S1E1WP is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of AT S1E1WP at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ATS1E1RP, bit [16]**When FEAT_PAN2 is implemented:**

Trap execution of [AT S1E1RP](#) at EL1 using AArch64 to EL2.

ATS1E1RP	Meaning
0b0	Execution of AT S1E1RP is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of AT S1E1RP at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ATS1E0W, bit [15]

Trap execution of [AT S1E0W](#) at EL1 using AArch64 to EL2.

ATS1E0W	Meaning
0b0	Execution of AT S1E0W is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of AT S1E0W at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ATS1E0R, bit [14]

Trap execution of [AT S1E0R](#) at EL1 using AArch64 to EL2.

ATS1E0R	Meaning
0b0	Execution of AT S1E0R is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of AT S1E0R at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ATS1E1W, bit [13]

Trap execution of [AT S1E1W](#) at EL1 using AArch64 to EL2.

ATS1E1W	Meaning
0b0	Execution of AT S1E1W is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of AT S1E1W at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ATS1E1R, bit [12]

Trap execution of [AT S1E1R](#) at EL1 using AArch64 to EL2.

ATS1E1R	Meaning
0b0	Execution of AT S1E1R is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution of AT S1E1R at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCZVA, bit [11]

Trap execution of multiple instructions. Enables a trap on execution at EL1 and EL0 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC ZVA](#).
- [DC GVA](#), if FEAT_MTE is implemented.
- [DC GZVA](#), if FEAT_MTE is implemented.

Note

Unlike [HCR_EL2.TDZ](#), this field has no effect on [DCZID_EL0.DZP](#).

DCZVA	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then execution at EL1 and EL0 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCCIVAC, bit [10]

Trap execution of multiple instructions. Enables a trap on execution at EL1 and EL0 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CIVAC](#).
- [DC CIGVAC](#), if FEAT_MTE is implemented.
- [DC CIGDVAC](#), if FEAT_MTE is implemented.

If the Point of Coherence is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCCIVAC	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 and EL0 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCCVADP, bit [9]

When FEAT_DPB2 is implemented:

Trap execution of multiple instructions. Enables a trap on execution at EL1 and EL0 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CVADP](#).
- [DC CGVADP](#), if FEAT_MTE is implemented.
- [DC CGDVADP](#), if FEAT_MTE is implemented.

If the Point of Deep Persistence is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCCVADP	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 and EL0 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

DCCVAP, bit [8]

Trap execution of multiple instructions. Enables a trap on execution at EL1 and EL0 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CVAP](#).
- [DC CGVAP](#), if FEAT_MTE is implemented.
- [DC CGDVAP](#), if FEAT_MTE is implemented.

If the Point of Persistence is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCCVAP	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 and EL0 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCCVAU, bit [7]

Trap execution of [DC CVAU](#) at EL1 and EL0 using AArch64 to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCCVAU	Meaning
0b0	Execution of DC CVAU is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of DC CVAU at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCCISW, bit [6]

Trap execution of multiple instructions. Enables a trap on execution at EL1 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CISW](#).
- [DC CIGSW](#), if FEAT_MTE2 is implemented.
- [DC CIGDSW](#), if FEAT_MTE2 is implemented.

DCCISW	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCCSW, bit [5]

Trap execution of multiple instructions. Enables a trap on execution at EL1 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC CSW](#).
- [DC CGSW](#), if FEAT_MTE2 is implemented.
- [DC CGDSW](#), if FEAT_MTE2 is implemented.

DCCSW	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCISW, bit [4]

Trap execution of multiple instructions. Enables a trap on execution at EL1 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC ISW](#).
- [DC IGSW](#), if FEAT_MTE2 is implemented.
- [DC IGDSW](#), if FEAT_MTE2 is implemented.

DCISW	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCIVAC, bit [3]

Trap execution of multiple instructions. Enables a trap on execution at EL1 using AArch64 of any of the following AArch64 instructions to EL2:

- [DC IVAC](#).
- [DC IGVAC](#), if FEAT_MTE2 is implemented.
- [DC IGDVAC](#), if FEAT_MTE2 is implemented.

If the Point of Coherence is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

DCIVAC	Meaning
0b0	Execution of the instructions listed above is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution at EL1 using AArch64 of any of the instructions listed above is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ICIVAU, bit [2]

Trap execution of [IC IVAU](#) at EL1 and EL0 using AArch64 to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

ICIVAU	Meaning
0b0	Execution of IC IVAU is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of IC IVAU at EL1 and EL0 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ICIALLU, bit [1]

Trap execution of [IC IALLU](#) at EL1 using AArch64 to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

ICIALLU	Meaning
0b0	Execution of IC IALLU is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of IC IALLU at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ICIALUIS, bit [0]

Trap execution of [IC IALLUIS](#) at EL1 using AArch64 to EL2.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of the affected instruction is trapped when the value of this control is 1.

ICIALUIS	Meaning
0b0	Execution of IC IALLUIS is not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then execution of IC IALLUIS at EL1 using AArch64 is trapped to EL2 and reported with EC syndrome value 0x18, unless the instruction generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing HFGITR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGITR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1C8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HFGITR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGITR_EL2;

```

MSR HFGITR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1C8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HFGITR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGITR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

HFGRTR2_EL2, Hypervisor Fine-Grained Read Trap Register 2

The HFGRTR2_EL2 characteristics are:

Purpose

Provides controls for traps of MRS and MRC reads of System registers.

Configuration

This register is present only when FEAT_FGT2 is implemented. Otherwise, direct accesses to HFGRTR2_EL2 are UNDEFINED.

Attributes

HFGRTR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:3]

Reserved, RES0.

nRCWSMASK_EL1, bit [2]

When FEAT_TME is implemented:

Trap MRS or MRRS reads of RCWSMASK_EL1 at EL1 using AArch64 to EL2.

nRCWSMASK_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MRS or MRRS reads of RCWSMASK_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18 for 64-bit access and 0x14 for 128-bit access, unless the read generates a higher priority exception.
0b1	MRS or MRRS reads of RCWSMASK_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nERXGSR_EL1, bit [1]**When FEAT_RASv2 is implemented:**

Trap MRS reads of [ERXGSR_EL1](#) at EL1 using AArch64 to EL2.

nERXGSR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads of ERXGSR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of ERXGSR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

nPFAR_EL1, bit [0]**When FEAT_PFAR is implemented:**

Trap MRS reads of [PFAR_EL1](#) at EL1 using AArch64 to EL2.

nPFAR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn2 == 1, then MRS reads of PFAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PFAR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing HFGRTR2_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGTR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2C0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HFGTR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGTR2_EL2;

```

MSR HFGTR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x2C0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HFGTR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGTR2_EL2 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

HFGRTR_EL2, Hypervisor Fine-Grained Read Trap Register

The HFGRTR_EL2 characteristics are:

Purpose

Provides controls for traps of MRS and MRC reads of System registers.

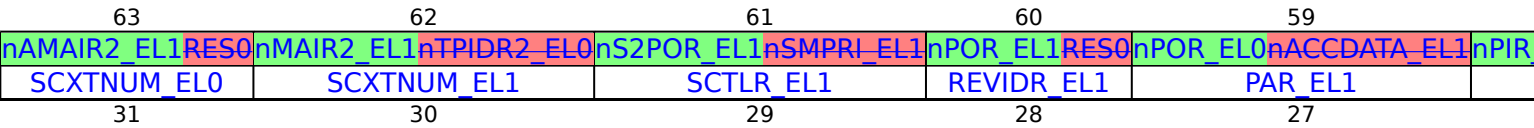
Configuration

This register is present only when FEAT_FGT is implemented. Otherwise, direct accesses to HFGRTR_EL2 are UNDEFINED.

Attributes

HFGRTR_EL2 is a 64-bit register.

Field descriptions



nMAIR2_EL1, Bits bit [63:56]
When FEAT_AIE is implemented:

Trap MRS reads of MAIR2_EL1 at EL1 using AArch64 to EL2.

nMAIR2_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MAIR2_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of MAIR2_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nMAIR2_EL1, bit [62]
When FEAT_AIE is implemented:

Trap MRS reads of MAIR2_EL1 at EL1 using AArch64 to EL2.

nMAIR2_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MAIR2_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of MAIR2_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nS2POR_EL1, bit [61]

When FEAT_S2POE is implemented:

Trap MRS reads of S2POR_EL1 at EL1 using AArch64 to EL2.

nS2POR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of S2POR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of S2POR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPOR_EL1, bit [60]

When FEAT_S1POE is implemented:

Trap MRS reads of POR_EL1 at EL1 using AArch64 to EL2.

nPOR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of POR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of POR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPOR_EL0, bit [59]**When FEAT_S1POE is implemented:**Trap MRS reads of [POR_EL0](#) at EL1 using AArch64 to EL2.

nPOR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of POR_EL0 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of POR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPIR_EL1, bit [58]**When FEAT_S1PIE is implemented:**Trap MRS reads of [PIR_EL1](#) at EL1 using AArch64 to EL2.

nPIR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PIR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPIRE0_EL1, bit [57]**When FEAT_S1PIE is implemented:**Trap MRS reads of [PIRE0_EL1](#) at EL1 using AArch64 to EL2.

nPIRE0_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of PIRE0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of PIRE0_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nRCWMASK_EL1, bit [56]

When FEAT_THE is implemented:

Trap MRS or MRRS reads of RCWMASK_EL1 at EL1 using AArch64 to EL2.

nRCWMASK_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS or MRRS reads of RCWMASK_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS or MRRS reads of RCWMASK_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nTPIDR2_EL0, bit [55]

When FEAT_SME is implemented:

Trap MRS reads of [TPIDR2_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nTPIDR2_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TPIDR2_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of TPIDR2_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nSMPRI_EL1, bit [54]**When FEAT_SME is implemented:**

Trap MRS reads of [SMPRI_EL1](#) at EL1 using AArch64 to EL2.

nSMPRI_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of SMPRI_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of SMPRI_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [53:51]

Reserved, RES0.

nACCDATA_EL1, bit [50]**When FEAT_LS64_ACCDATA is implemented:**

Trap MRS reads of [ACCDATA_EL1](#) at EL1 using AArch64 to EL2.

nACCDATA_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ACCDATA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.
0b1	MRS reads of ACCDATA_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXADDR_EL1, bit [49]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERXADDR_EL1](#) at EL1 using AArch64 to EL2.

ERXADDR_EL1	Meaning
0b0	MRS reads of ERXADDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXADDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXPFPCDN_EL1, bit [48]

When FEAT_RASv1p1 is implemented:

Trap MRS reads of [ERXPFPCDN_EL1](#) at EL1 using AArch64 to EL2.

ERXPFPCDN_EL1	Meaning
0b0	MRS reads of ERXPFPCDN_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXPFPCDN_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXPFCTL_EL1, bit [47]

When FEAT_RASv1p1 is implemented:

Trap MRS reads of [ERXPFCTL_EL1](#) at EL1 using AArch64 to EL2.

ERXPFCTL_EL1	Meaning
0b0	MRS reads of ERXPFCTL_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXPFCTL_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXPFGF_EL1, bit [46]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERXPFGF_EL1](#) at EL1 using AArch64 to EL2.

ERXPFGF_EL1	Meaning
0b0	MRS reads of ERXPFGF_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXPFGF_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXMISCN_EL1, bit [45]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERXMISCN_EL1](#) at EL1 using AArch64 to EL2.

ERXMISCN_EL1	Meaning
0b0	MRS reads of ERXMISCN_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXMISCN_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXSTATUS_EL1, bit [44]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERXSTATUS_EL1](#) at EL1 using AArch64 to EL2.

ERXSTATUS_EL1	Meaning
0b0	MRS reads of ERXSTATUS_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXSTATUS_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXCTLR_EL1, bit [43]

When FEAT_RAS is implemented:

Trap MRS reads of [ERXCTLR_EL1](#) at EL1 using AArch64 to EL2.

ERXCTLR_EL1	Meaning
0b0	MRS reads of ERXCTLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXCTLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXFR_EL1, bit [42]

When FEAT_RAS is implemented:

Trap MRS reads of [ERXFR_EL1](#) at EL1 using AArch64 to EL2.

ERXFR_EL1	Meaning
0b0	MRS reads of ERXFR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERXFR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERRSEL_EL1, bit [41]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERRSEL_EL1](#) at EL1 using AArch64 to EL2.

ERRSEL_EL1	Meaning
0b0	MRS reads of ERRSEL_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERRSEL_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERRIDR_EL1, bit [40]**When FEAT_RAS is implemented:**

Trap MRS reads of [ERRIDR_EL1](#) at EL1 using AArch64 to EL2.

ERRIDR_EL1	Meaning
0b0	MRS reads of ERRIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ERRIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ICC_IGRPENn_EL1, bit [39]**When FEAT_GICv3 is implemented:**

Trap MRS reads of ICC_IGRPEN<n>_EL1 at EL1 using AArch64 to EL2.

ICC_IGRPEN _n _EL1	Meaning
0b0	MRS reads of ICC_IGRPEN<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ICC_IGRPEN<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

VBAR_EL1, bit [38]

Trap MRS reads of [VBAR_EL1](#) at EL1 using AArch64 to EL2.

VBAR_EL1	Meaning
0b0	MRS reads of VBAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of VBAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TTBR1_EL1, bit [37]

Trap MRS ~~reads of~~ [MRRS reads of](#) [TTBR1_EL1](#) at EL1 using AArch64 to EL2.

TTBR1_EL1	Meaning
0b0	MRS reads of MRRS reads of TTBR1_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MRRS reads of TTBR1_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TTBR0_EL1, bit [36]

Trap MRS ~~reads of~~ [MRRS reads of](#) [TTBR0_EL1](#) at EL1 using AArch64 to EL2.

TTBR0_EL1	Meaning
0b0	MRS or reads of MRRS reads of TTBR0_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS or reads of MRRS reads of TTBR0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDR_EL0, bit [35]

Trap MRS reads of [TPIDR_EL0](#) at EL1 and EL0 using AArch64 and MRC reads of [TPIDRURW](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

TPIDR_EL0	Meaning
0b0	MRS reads of TPIDR_EL0 at EL1 and EL0 using AArch64 and MRC reads of TPIDRURW at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> MRS reads of TPIDR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads of TPIDRURW at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDRRO_EL0, bit [34]

Trap MRS reads of [TPIDRRO_EL0](#) at EL1 and EL0 using AArch64 and MRC reads of [TPIDRURO](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

TPIDRRO_EL0	Meaning
0b0	MRS reads of TPIDRRO_EL0 at EL1 and EL0 using AArch64 and MRC reads of TPIDRURO at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the read generates a higher priority exception: <ul style="list-style-type: none"> MRS reads of TPIDRRO_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MRC reads of TPIDRURO at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDR_EL1, bit [33]

Trap MRS reads of [TPIDR_EL1](#) at EL1 using AArch64 to EL2.

TPIDR_EL1	Meaning
0b0	MRS reads of TPIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of TPIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TCR_EL1, bit [32]

Trap MRS reads of [any of the following registers at EL1 using AArch64 to EL2](#). [TCR_EL1](#) at EL1 using AArch64 to [EL2](#).

- [TCR_EL1](#).
- [TCR2_EL1](#), if FEAT_TCR2 is implemented.

TCR_EL1	Meaning
0b0	MRS reads of the specified registers are not trapped by this mechanism . TCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of the specified registers at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value TCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

SCXTNUM_ELO, bit [31]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Trap MRS reads of [SCXTNUM_ELO](#) at EL1 and EL0 using AArch64 to EL2.

SCXTNUM_ELO	Meaning
0b0	MRS reads of SCXTNUM_ELO are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of SCXTNUM_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

SCXTNUM_EL1, bit [30]**When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:**

Trap MRS reads of [SCXTNUM_EL1](#) at EL1 using AArch64 to EL2.

SCXTNUM_EL1	Meaning
0b0	MRS reads of SCXTNUM_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of SCXTNUM_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

SCTLR_EL1, bit [29]

Trap MRS reads of any of the following registers at EL1 using AArch64 to EL2. [SCTLR_EL1](#) at EL1 using AArch64 to EL2.

- [SCTLR_EL1](#).
- [SCTLR2_EL1](#), if FEAT_SCTLR2 is implemented.

SCTLR_EL1	Meaning
0b0	MRS reads of the specified registers are not trapped by this mechanism. SCTLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of the specified registers at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value SCTLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

REVIDR_EL1, bit [28]

Trap MRS reads of [REVIDR_EL1](#) at EL1 using AArch64 to EL2.

REVIDR_EL1	Meaning
0b0	MRS reads of REVIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of REVIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

PAR_EL1, bit [27]

Trap MRS ~~or reads of~~ MRRS reads of [PAR_EL1](#) at EL1 using AArch64 to EL2.

PAR_EL1	Meaning
0b0	MRS or reads of MRRS reads of PAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS or reads of MRRS reads of PAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

MPIDR_EL1, bit [26]

Trap MRS reads of [MPIDR_EL1](#) at EL1 using AArch64 to EL2.

MPIDR_EL1	Meaning
0b0	MRS reads of MPIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MPIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

MIDR_EL1, bit [25]

Trap MRS reads of [MIDR_EL1](#) at EL1 using AArch64 to EL2.

MIDR_EL1	Meaning
0b0	MRS reads of MIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

MAIR_EL1, bit [24]

Trap MRS reads of [MAIR_EL1](#) at EL1 using AArch64 to EL2.

MAIR_EL1	Meaning
0b0	MRS reads of MAIR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of MAIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

LORSA_EL1, bit [23]

When FEAT_LOR is implemented:

Trap MRS reads of [LORSA_EL1](#) at EL1 using AArch64 to EL2.

LORSA_EL1	Meaning
0b0	MRS reads of LORSA_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of LORSA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LORN_EL1, bit [22]

When FEAT_LOR is implemented:

Trap MRS reads of [LORN_EL1](#) at EL1 using AArch64 to EL2.

LORN_EL1	Meaning
0b0	MRS reads of LORN_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of LORN_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LORID_EL1, bit [21]

When FEAT_LOR is implemented:

Trap MRS reads of [LORID_EL1](#) at EL1 using AArch64 to EL2.

LORID_EL1	Meaning
0b0	MRS reads of LORID_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of LORID_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LOREA_EL1, bit [20]

When FEAT_LOR is implemented:

Trap MRS reads of [LOREA_EL1](#) at EL1 using AArch64 to EL2.

LOREA_EL1	Meaning
0b0	MRS reads of LOREA_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of LOREA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LORC_EL1, bit [19]

When FEAT_LOR is implemented:

Trap MRS reads of [LORC_EL1](#) at EL1 using AArch64 to EL2.

LORC_EL1	Meaning
0b0	MRS reads of LORC_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of LORC_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ISR_EL1, bit [18]

Trap MRS reads of [ISR_EL1](#) at EL1 using AArch64 to EL2.

ISR_EL1	Meaning
0b0	MRS reads of ISR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ISR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

FAR_EL1, bit [17]

Trap MRS reads of [FAR_EL1](#) at EL1 using AArch64 to EL2.

FAR_EL1	Meaning
0b0	MRS reads of FAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of FAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ESR_EL1, bit [16]

Trap MRS reads of [ESR_EL1](#) at EL1 using AArch64 to EL2.

ESR_EL1	Meaning
0b0	MRS reads of ESR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of ESR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

DCZID_EL0, bit [15]

Trap MRS reads of [DCZID_EL0](#) at EL1 and EL0 using AArch64 to EL2.

DCZID_EL0	Meaning
0b0	MRS reads of DCZID_EL0 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then MRS reads of DCZID_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CTR_EL0, bit [14]

Trap MRS reads of [CTR_EL0](#) at EL1 and EL0 using AArch64 to EL2.

CTR_EL0	Meaning
0b0	MRS reads of CTR_EL0 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then MRS reads of CTR_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CSSELR_EL1, bit [13]

Trap MRS reads of [CSSELR_EL1](#) at EL1 using AArch64 to EL2.

CSSELR_EL1	Meaning
0b0	MRS reads of CSSELR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then MRS reads of CSSELR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CPACR_EL1, bit [12]

Trap MRS reads of [CPACR_EL1](#) at EL1 using AArch64 to EL2.

CPACR_EL1	Meaning
0b0	MRS reads of CPACR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTEn == 1, then MRS reads of CPACR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CONTEXTIDR_EL1, bit [11]

Trap MRS reads of [CONTEXTIDR_EL1](#) at EL1 using AArch64 to EL2.

CONTEXTIDR_EL1	Meaning
0b0	MRS reads of CONTEXTIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of CONTEXTIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CLIDR_EL1, bit [10]

Trap MRS reads of [CLIDR_EL1](#) at EL1 using AArch64 to EL2.

CLIDR_EL1	Meaning
0b0	MRS reads of CLIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of CLIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CCSIDR_EL1, bit [9]

Trap MRS reads of [CCSIDR_EL1](#) at EL1 using AArch64 to EL2.

CCSIDR_EL1	Meaning
0b0	MRS reads of CCSIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of CCSIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

APIBKey, bit [8]**When FEAT_PAAuth is implemented:**

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APIBKeyHi_EL1](#).
- [APIBKeyLo_EL1](#).

APIBKey	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APIAKey, bit [7]

When FEAT_PAAuth is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APIAKeyHi_EL1](#).
- [APIAKeyLo_EL1](#).

APIAKey	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APGAKey, bit [6]

When FEAT_PAAuth is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APGAKeyHi_EL1](#).
- [APGAKeyLo_EL1](#).

APGAKey	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APDBKey, bit [5]

When FEAT_PAAuth is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APDBKeyHi_EL1](#).
- [APDBKeyLo_EL1](#).

APDBKey	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APDAKey, bit [4]

When FEAT_PAAuth is implemented:

Trap MRS reads of multiple System registers. Enables a trap on MRS reads at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APDAKeyHi_EL1](#).
- [APDAKeyLo_EL1](#).

APDAKey	Meaning
0b0	MRS reads of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

AMAIR_EL1, bit [3]

Trap MRS reads of [AMAIR_EL1](#) at EL1 using AArch64 to EL2.

AMAIR_EL1	Meaning
0b0	MRS reads of AMAIR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of AMAIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

AIDR_EL1, bit [2]

Trap MRS reads of [AIDR_EL1](#) at EL1 using AArch64 to EL2.

AIDR_EL1	Meaning
0b0	MRS reads of AIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of AIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

AFSR1_EL1, bit [1]

Trap MRS reads of [AFSR1_EL1](#) at EL1 using AArch64 to EL2.

AFSR1_EL1	Meaning
0b0	MRS reads of AFSR1_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of AFSR1_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

AFSR0_EL1, bit [0]

Trap MRS reads of [AFSR0_EL1](#) at EL1 using AArch64 to EL2.

AFSR0_EL1	Meaning
0b0	MRS reads of AFSR0_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MRS reads of AFSR0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the read generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing HFGTR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1B8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HFGTR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGTR_EL2;

```

MSR HFGRTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1B8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HFGRTR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGRTR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Otherwise:

Reserved, RES0.

Bit [1]

Reserved, RES0.

nPFAR_EL1, bit [0]**When FEAT_PFAR is implemented:**

Trap MSR writes of [PFAR_EL1](#) at EL1 using AArch64 to EL2.

nPFAR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn2 == 1, then MSR writes of PFAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PFAR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing HFGWTR2_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGWTR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2C8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HFGWTR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGWTR2_EL2;

```

MSR HFGWTR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0011	0b0001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x2C8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HFGWTR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGWTR2_EL2 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HFGWTR_EL2, Hypervisor Fine-Grained Write Trap Register

The HFGWTR_EL2 characteristics are:

Purpose

Provides controls for traps of MSR and MCR writes of System registers.

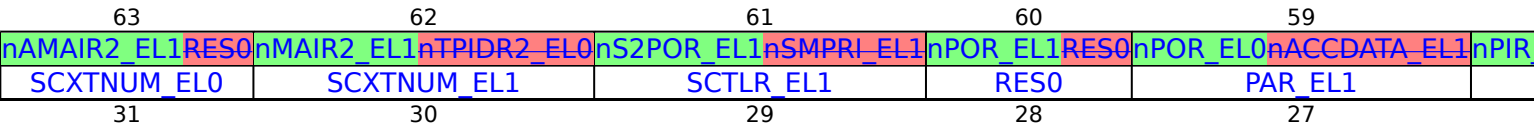
Configuration

This register is present only when FEAT_FGT is implemented. Otherwise, direct accesses to HFGWTR_EL2 are UNDEFINED.

Attributes

HFGWTR_EL2 is a 64-bit register.

Field descriptions



nMAIR2_EL1, Bits bit [63:56]
When FEAT_AIE is implemented:

Trap MSR writes of MAIR2_EL1 at EL1 using AArch64 to EL2.

nMAIR2_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of MAIR2_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of MAIR2_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nMAIR2_EL1, bit [62]
When FEAT_AIE is implemented:

Trap MSR writes of MAIR2_EL1 at EL1 using AArch64 to EL2.

nMAIR2_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of MAIR2_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of MAIR2_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nS2POR_EL1, bit [61]

When FEAT_S2POE is implemented:

Trap MSR writes of S2POR_EL1 at EL1 using AArch64 to EL2.

nS2POR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of S2POR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of S2POR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPOR_EL1, bit [60]

When FEAT_S1POE is implemented:

Trap MSR writes of POR_EL1 at EL1 using AArch64 to EL2.

nPOR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of POR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of POR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPOR_EL0, bit [59]**When FEAT_S1POE is implemented:**Trap MSR writes of [POR_EL0](#) at EL1 using AArch64 to EL2.

nPOR_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of POR_EL0 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of POR_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPIR_EL1, bit [58]**When FEAT_S1PIE is implemented:**Trap MSR writes of [PIR_EL1](#) at EL1 using AArch64 to EL2.

nPIR_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PIR_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nPIRE0_EL1, bit [57]**When FEAT_S1PIE is implemented:**Trap MSR writes of [PIRE0_EL1](#) at EL1 using AArch64 to EL2.

nPIRE0_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of PIRE0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of PIRE0_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nRCWMASK_EL1, bit [56]

When FEAT_THE is implemented:

Trap MSR or MSRR writes of RCWMASK_EL1 at EL1 using AArch64 to EL2.

nRCWMASK_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of RCWMASK_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of RCWMASK_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nTPIDR2_EL0, bit [55]

When FEAT_SME is implemented:

Trap MSR writes of [TPIDR2_EL0](#) at EL1 and EL0 using AArch64 to EL2.

nTPIDR2_EL0	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TPIDR2_EL0 at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of TPIDR2_EL0 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

nSMPRI_EL1, bit [54]**When FEAT_SME is implemented:**

Trap MSR writes of [SMPRI_EL1](#) at EL1 using AArch64 to EL2.

nSMPRI_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of SMPRI_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of SMPRI_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [53:51]

Reserved, RES0.

nACCDATA_EL1, bit [50]**When FEAT_LS64_ACCDATA is implemented:**

Trap MSR writes of [ACCDATA_EL1](#) at EL1 using AArch64 to EL2.

nACCDATA_EL1	Meaning
0b0	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ACCDATA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.
0b1	MSR writes of ACCDATA_EL1 are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXADDR_EL1, bit [49]**When FEAT_RAS is implemented:**

Trap MSR writes of [ERXADDR_EL1](#) at EL1 using AArch64 to EL2.

ERXADDR_EL1	Meaning
0b0	MSR writes of ERXADDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ERXADDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXPFPGCDN_EL1, bit [48]

When FEAT_RASv1p1 is implemented:

Trap MSR writes of [ERXPFPGCDN_EL1](#) at EL1 using AArch64 to EL2.

ERXPFPGCDN_EL1	Meaning
0b0	MSR writes of ERXPFPGCDN_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ERXPFPGCDN_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXPFPGCTL_EL1, bit [47]

When FEAT_RASv1p1 is implemented:

Trap MSR writes of [ERXPFPGCTL_EL1](#) at EL1 using AArch64 to EL2.

ERXPFPGCTL_EL1	Meaning
0b0	MSR writes of ERXPFPGCTL_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ERXPFPGCTL_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [46]

Reserved, RES0.

ERXMISCn_EL1, bit [45]**When FEAT_RAS is implemented:**

Trap MSR writes of ERXMISC<n>_EL1 at EL1 using AArch64 to EL2.

ERXMISCn_EL1	Meaning
0b0	MSR writes of ERXMISC<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ERXMISC<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXSTATUS_EL1, bit [44]**When FEAT_RAS is implemented:**

Trap MSR writes of [ERXSTATUS_EL1](#) at EL1 using AArch64 to EL2.

ERXSTATUS_EL1	Meaning
0b0	MSR writes of ERXSTATUS_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ERXSTATUS_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

ERXCTLR_EL1, bit [43]**When FEAT_RAS is implemented:**

Trap MSR writes of [ERXCTLR_EL1](#) at EL1 using AArch64 to EL2.

ERXCTLR_EL1	Meaning
0b0	MSR writes of ERXCTLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MSR writes of ERXCTLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [42]

Reserved, RES0.

ERRSELR_EL1, bit [41]**When FEAT_RAS is implemented:**

Trap MSR writes of [ERRSELR_EL1](#) at EL1 using AArch64 to EL2.

ERRSELR_EL1	Meaning
0b0	MSR writes of ERRSELR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3 .FGTE _n == 1, then MSR writes of ERRSELR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [40]

Reserved, RES0.

ICC_IGRPEN_n_EL1, bit [39]**When FEAT_GICv3 is implemented:**

Trap MSR writes of ICC_IGRPEN<n>_EL1 at EL1 using AArch64 to EL2.

ICC_IGRPEN _n _EL1	Meaning
0b0	MSR writes of ICC_IGRPEN<n>_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ICC_IGRPEN<n>_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

VBAR_EL1, bit [38]

Trap MSR ~~or writes of~~ MSRR writes of [VBAR_EL1](#) at EL1 using AArch64 to EL2.

VBAR_EL1	Meaning
0b0	MSR or writes of MSRR writes of VBAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR or writes of MSRR writes of VBAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TTBR1_EL1, bit [37]

Trap MSR ~~or writes of~~ MSRR writes of [TTBR1_EL1](#) at EL1 using AArch64 to EL2.

TTBR1_EL1	Meaning
0b0	MSR or writes of MSRR writes of TTBR1_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR or writes of MSRR writes of TTBR1_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TTBR0_EL1, bit [36]

Trap MSR ~~or writes of~~ MSRR writes of [TTBR0_EL1](#) at EL1 using AArch64 to EL2.

TTBR0_EL1	Meaning
0b0	MSR or writes of MSRR writes of TTBR0_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR or writes of MSRR writes of TTBR0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDR_ELO, bit [35]

Trap MSR writes of [TPIDR_ELO](#) at EL1 and EL0 using AArch64 and MCR writes of [TPIDRURW](#) at EL0 using AArch32 when EL1 is using AArch64 to EL2.

TPIDR_ELO	Meaning
0b0	MSR writes of TPIDR_ELO at EL1 and EL0 using AArch64 and MCR writes of TPIDRURW at EL0 using AArch32 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2.{E2H, TGE} != {1, 1}, EL1 is using AArch64, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then, unless the write generates a higher priority exception: <ul style="list-style-type: none"> MSR writes of TPIDR_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18. MCR writes of TPIDRURW at EL0 using AArch32 are trapped to EL2 and reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDRRO_ELO, bit [34]

Trap MSR writes of [TPIDRRO_ELO](#) at EL1 using AArch64 to EL2.

TPIDRRO_ELO	Meaning
0b0	MSR writes of TPIDRRO_ELO are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TPIDRRO_ELO at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TPIDR_EL1, bit [33]

Trap MSR writes of [TPIDR_EL1](#) at EL1 using AArch64 to EL2.

TPIDR_EL1	Meaning
0b0	MSR writes of TPIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of TPIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

TCR_EL1, bit [32]

Trap MSR writes of any of the following registers at EL1 using AArch64 to EL2. [TCR_EL1](#) at EL1 using AArch64 to EL2.

- [TCR_EL1](#).
- [TCR2_EL1](#), if FEAT_TCR2 is implemented.

TCR_EL1	Meaning
0b0	MSR writes of the specified registers are not trapped by this mechanism. TCR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of the specified registers at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value TCR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

SCXTNUM_ELO, bit [31]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Trap MSR writes of [SCXTNUM_ELO](#) at EL1 and EL0 using AArch64 to EL2.

SCXTNUM_ELO	Meaning
0b0	MSR writes of SCXTNUM_ELO are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, HCR_EL2 .{E2H, TGE} != {1, 1}, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of SCXTNUM_ELO at EL1 and EL0 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

SCXTNUM_EL1, bit [30]**When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:**

Trap MSR writes of [SCXTNUM_EL1](#) at EL1 using AArch64 to EL2.

SCXTNUM_EL1	Meaning
0b0	MSR writes of SCXTNUM_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of SCXTNUM_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

SCTLR_EL1, bit [29]

Trap MSR writes of any of the following registers at EL1 using AArch64 to EL2. [SCTLR_EL1](#) at EL1 using AArch64 to EL2.

- [SCTLR_EL1](#).
- [SCTLR2_EL1](#), if FEAT_SCTLR2 is implemented.

SCTLR_EL1	Meaning
0b0	MSR writes of the specified registers are not trapped by this mechanism. SCTLR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of the specified registers at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value SCTLR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bit [28]

Reserved, RES0.

PAR_EL1, bit [27]

Trap MSR or writes of MSRR writes of [PAR_EL1](#) at EL1 using AArch64 to EL2.

PAR_EL1	Meaning
0b0	MSR or writes of MSRR writes of PAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR or writes of MSRR writes of PAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bits [26:25]

Reserved, RES0.

MAIR_EL1, bit [24]

Trap MSR writes of [MAIR_EL1](#) at EL1 using AArch64 to EL2.

MAIR_EL1	Meaning
0b0	MSR writes of MAIR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of MAIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

LORSA_EL1, bit [23]

When FEAT_LOR is implemented:

Trap MSR writes of [LORSA_EL1](#) at EL1 using AArch64 to EL2.

LORSA_EL1	Meaning
0b0	MSR writes of LORSA_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of LORSA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LORN_EL1, bit [22]

When FEAT_LOR is implemented:

Trap MSR writes of [LORN_EL1](#) at EL1 using AArch64 to EL2.

LORN_EL1	Meaning
0b0	MSR writes of LORN_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of LORN_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [21]

Reserved, RES0.

LOREA_EL1, bit [20]

When FEAT_LOR is implemented:

Trap MSR writes of [LOREA_EL1](#) at EL1 using AArch64 to EL2.

LOREA_EL1	Meaning
0b0	MSR writes of LOREA_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of LOREA_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

LORC_EL1, bit [19]

When FEAT_LOR is implemented:

Trap MSR writes of [LORC_EL1](#) at EL1 using AArch64 to EL2.

LORC_EL1	Meaning
0b0	MSR writes of LORC_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of LORC_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [18]

Reserved, RES0.

FAR_EL1, bit [17]

Trap MSR writes of [FAR_EL1](#) at EL1 using AArch64 to EL2.

FAR_EL1	Meaning
0b0	MSR writes of FAR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of FAR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

ESR_EL1, bit [16]

Trap MSR writes of [ESR_EL1](#) at EL1 using AArch64 to EL2.

ESR_EL1	Meaning
0b0	MSR writes of ESR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of ESR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bits [15:14]

Reserved, RES0.

CSSELR_EL1, bit [13]

Trap MSR writes of [CSSELR_EL1](#) at EL1 using AArch64 to EL2.

CSSELR_EL1	Meaning
0b0	MSR writes of CSSELR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of CSSELR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CPACR_EL1, bit [12]

Trap MSR writes of [CPACR_EL1](#) at EL1 using AArch64 to EL2.

CPACR_EL1	Meaning
0b0	MSR writes of CPACR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of CPACR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

CONTEXTIDR_EL1, bit [11]

Trap MSR writes of [CONTEXTIDR_EL1](#) at EL1 using AArch64 to EL2.

CONTEXTIDR_EL1	Meaning
0b0	MSR writes of CONTEXTIDR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of CONTEXTIDR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bits [10:9]

Reserved, RES0.

APIBKey, bit [8]

When FEAT_PAuth is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APIBKeyHi_EL1](#).
- [APIBKeyLo_EL1](#).

APIBKey	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APIAKey, bit [7]

When FEAT_PAAuth is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APIAKeyHi_EL1](#).
- [APIAKeyLo_EL1](#).

APIAKey	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APGAKey, bit [6]

When FEAT_PAAuth is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APGAKeyHi_EL1](#).
- [APGAKeyLo_EL1](#).

APGAKey	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APDBKey, bit [5]

When FEAT_PAAuth is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APDBKeyHi_EL1](#).
- [APDBKeyLo_EL1](#).

APDBKey	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

APDAKey, bit [4]

When FEAT_PAAuth is implemented:

Trap MSR writes of multiple System registers. Enables a trap on MSR writes at EL1 using AArch64 of any of the following AArch64 System registers to EL2:

- [APDAKeyHi_EL1](#).
- [APDAKeyLo_EL1](#).

APDAKey	Meaning
0b0	MSR writes of the System registers listed above are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes at EL1 using AArch64 of any of the System registers listed above are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

AMAIR_EL1, bit [3]

Trap MSR writes of [AMAIR_EL1](#) at EL1 using AArch64 to EL2.

AMAIR_EL1	Meaning
0b0	MSR writes of AMAIR_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of AMAIR_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Bit [2]

Reserved, RES0.

AFSR1_EL1, bit [1]

Trap MSR writes of [AFSR1_EL1](#) at EL1 using AArch64 to EL2.

AFSR1_EL1	Meaning
0b0	MSR writes of AFSR1_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of AFSR1_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

AFSR0_EL1, bit [0]

Trap MSR writes of [AFSR0_EL1](#) at EL1 using AArch64 to EL2.

AFSR0_EL1	Meaning
0b0	MSR writes of AFSR0_EL1 are not trapped by this mechanism.
0b1	If EL2 is implemented and enabled in the current Security state, and either EL3 is not implemented or SCR_EL3.FGTEn == 1, then MSR writes of AFSR0_EL1 at EL1 using AArch64 are trapped to EL2 and reported with EC syndrome value 0x18, unless the write generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing HFGWTR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, HFGWTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x1C0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = HFGWTR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HFGWTR_EL2;

```

MSR HFGWTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1C0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FGTEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.FGTEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        HFGWTR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HFGWTR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HPFAR_EL2, Hypervisor IPA Fault Address Register

The HPFAR_EL2 characteristics are:

Purpose

Holds the faulting IPA for some aborts on a stage 2 translation taken to EL2.

Configuration

AArch64 System register HPFAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HPFAR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

The HPFAR_EL2 is written for:

- Translation or Access faults in the second stage of translation.
- An abort in the second stage of translation performed during the translation table walk of a first stage translation, caused by a Translation fault, an Access flag fault, or a Permission fault.
- A stage 2 Address size fault.
- If FEAT_RME is implemented, a Granule Protection Check fault in the second stage of translation.

For all other exceptions taken to EL2, this register is UNKNOWN.

Note

The address held in this register is an address accessed by the instruction fetch or data access that caused the exception that gave rise to the Instruction Abort exception or Data Abort exception. It is the lowest address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

Attributes

HPFAR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NS		RES0														FIPA															
FIPA																								RES0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Execution at EL1 or EL0 makes HPFAR_EL2 become UNKNOWN.

NS, bit [63]

When FEAT_SEL2 is implemented:

Faulting IPA address space.

NS	Meaning
0b0	Faulting IPA is from the Secure IPA space.
0b1	Faulting IPA is from the Non-secure IPA space.

For Data Abort exceptions or Instruction Abort exceptions taken to Non-secure EL2:

- This field is RES0.
- The address is from the Non-secure IPA space.

If FEAT_RME is implemented, for Data Abort exceptions or Instruction Abort exceptions taken to Realm EL2:

- This field is RES0.
- The address is from the Realm IPA space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

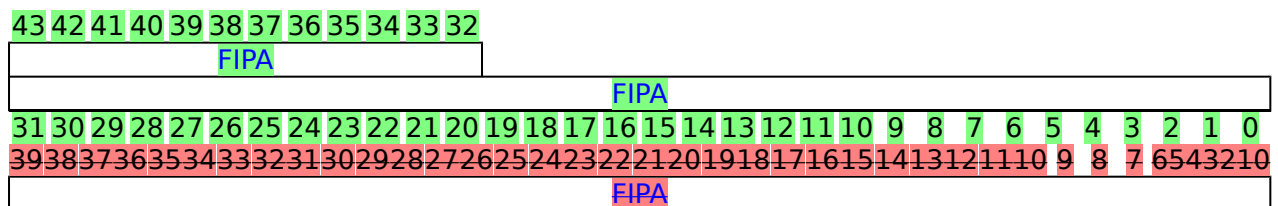
Reserved, RES0.

Bits [62:48:44]

Reserved, RES0.

FIPA, bits [47:43:4]

FIPA encoding when FEAT_D128FEAT_LPA is implemented



FIPA, bits [43:39:0]

Bits [55:4:12] of the Faulting Intermediate Physical Address.

For implementations with fewer than 55:52 physical address bits, the corresponding upper bits in this field are RES0.

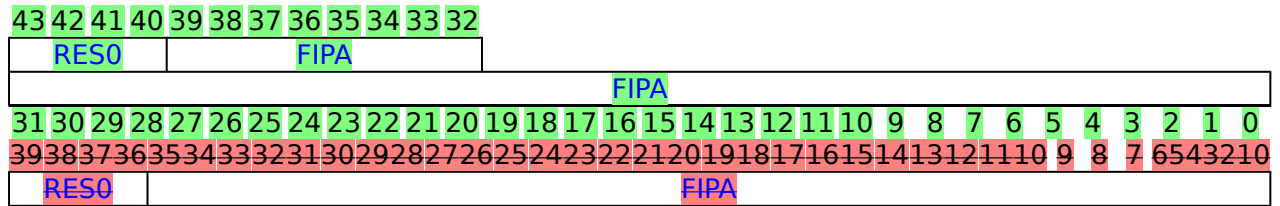
When FEAT_MOPS is implemented, the value presented in FIPA on a synchronous exception that set the HPFAR_EL2 from any of the Memory Copy and Memory Set instructions is within the address range of the current stage 2 translation granule, aligned to the size of the current stage 2 translation granule, of the address that generated the Data abort.

Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the current stage 2 translation granule size in bytes.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FIPA encoding when FEAT_LPA is implemented and FEAT_D128 is not implemented



Bits [43:39]:40:36]

Reserved, RES0.

FIPA, bits [39:35]:0]

Bits [51:47]:12] of the Faulting Intermediate Physical Address.

For implementations with fewer than 52:48 physical address bits, the corresponding upper bits in this field are RES0.

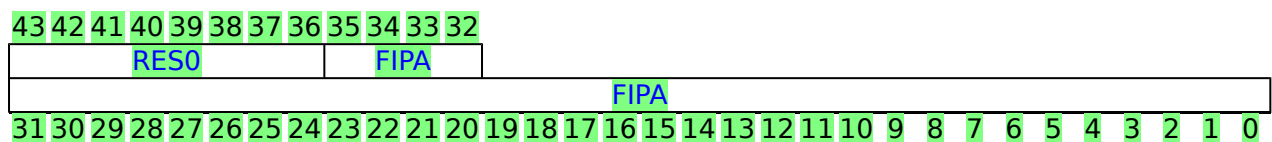
When FEAT_MOPS is implemented, the value presented in FIPA on a synchronous exception that set the HPFAR_EL2 from any of the Memory Copy and Memory Set instructions is within the address range of the current stage 2 translation granule, aligned to the size of the current stage 2 translation granule, of the address that generated the Data abort.

Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the current stage 2 translation granule size in bytes.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FIPA encoding when FEAT_LPA is not implemented



Bits [43:36]

Reserved, RES0.

FIPA, bits [35:0]

Bits[47:12] Faulting Intermediate Physical Address.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

When FEAT_MOPS is implemented, the value presented in FIPA on a synchronous exception that set the HPFAR_EL2 from any of the Memory Copy and Memory Set instructions is within the address range of the current stage 2 translation granule, aligned to the size of the current stage 2 translation granule, of the address that generated the Data abort.

Bits[(n-1):0] of the value are UNKNOWN, where 2^n is the current stage 2 translation granule size in bytes.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

(old)

htmldiff from-

(new)

IC IALLU, Instruction Cache Invalidate All to PoU

The IC IALLU characteristics are:

Purpose

Invalidate all instruction caches of the PE executing the instruction to the Point of Unification.

Configuration

AArch64 System instruction IC IALLU performs the same function as AArch32 System instruction [ICIALLU](#).

Attributes

IC IALLU is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the IC IALLU instruction

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

IC IALLU{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TPU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TOCU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.ICIALLU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.IC(CacheOpScope_ALLUIS);
    else
        AArch64.IC(CacheOpScope_ALLU);
elsif PSTATE.EL == EL2 then
    AArch64.IC(CacheOpScope_ALLU);
elsif PSTATE.EL == EL3 then
    AArch64.IC(CacheOpScope_ALLU);

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TPU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TICAB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ICIALLUIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.IC(CacheOpScope_ALLUIS);
elsif PSTATE.EL == EL2 then
    AArch64.IC(CacheOpScope_ALLUIS);
elsif PSTATE.EL == EL3 then
    AArch64.IC(CacheOpScope_ALLUIS);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

IC IVAU, Instruction Cache line Invalidate by VA to PoU

The IC IVAU characteristics are:

Purpose

Invalidate instruction cache by address to Point of Unification.

Configuration

AArch64 System instruction IC IVAU performs the same function as AArch32 System instruction [ICIMVAU](#).

Attributes

IC IVAU is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the IC IVAU instruction

If EL0 access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The [data instruction](#) cache maintenance instruction (DCIC)'.
~~If EL0 access is enabled, when executed at EL0, if this instruction does not have read access permission to the VA, it is IMPLEMENTATION DEFINED whether it generates a Permission fault.~~

~~If EL0 access is enabled, when executed at EL0, if this instruction does not have read access permission to the VA, it is IMPLEMENTATION DEFINED whether it generates a Permission fault.~~

For more information, see 'Permission fault'.

Accesses to this instruction use the following encodings in the System instruction encoding space:

IC IVAU{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0101	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TOCU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ICIVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.TOCU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.ICIVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
        AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL3 then
        AArch64.IC(X[t, 64], CacheOpScope_PoU);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_AP1R<n>_EL1, Interrupt Controller Active Priorities Group 1 Registers, n = 0 - 3

The ICC_AP1R<n>_EL1 characteristics are:

Purpose

Provides information about Group 1 active priorities.

Configuration

AArch64 System register ICC_AP1R<n>_EL1 bits [31:0] (ICC_AP1R<n>_EL1_S) are architecturally mapped to AArch32 System register [ICC_AP1R<n>\[31:0\]](#) (ICC_AP1R<n>_S).

AArch64 System register ICC_AP1R<n>_EL1 bits [31:0] (ICC_AP1R<n>_EL1_NS) are architecturally mapped to AArch32 System register [ICC_AP1R<n>\[31:0\]](#) (ICC_AP1R<n>_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_AP1R<n>_EL1 are UNDEFINED.

Attributes

ICC_AP1R<n>_EL1 is a 64-bit register.

This register has the following instances:

- ICC_AP1R<n>_EL1, when when EL3 is not implemented
- ICC_AP1R<n>_EL1_S, when when EL3 is implemented
- ICC_AP1R<n>_EL1_NS, when when EL3 is implemented

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NMI		RES0																													
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NMI, bit [63]

When FEAT_GICv3_NMI is implemented and n == 0:

Indicates whether there is an active NMI priority.

NMI	Meaning
0b0	There is no active Group 1 NMI, or all active Group 1 NMIs have undergone priority drop.
0b1	There is an active Group 1 NMI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [62:32]

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Accessing ICC_AP1R<n>_EL1

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are UNDEFINED.

Note

The number of bits of preemption is indicated by [ICH_VTR_EL2](#).PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- [ICC_AP0R<n>_EL1](#).
- Secure ICC_AP1R<n>_EL1.
- Non-secure ICC_AP1R<n>_EL1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_AP1R<m>_EL1 ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(op2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elsif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        X[t, 64] = ICV_AP1R_EL1[m];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[m];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[m];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[m];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[m];

```

MSR ICC_AP1R<m>_EL1, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(op2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elsif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[m] = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_AP1R_EL1_S[m] = X[t, 64];
            else
                ICC_AP1R_EL1_NS[m] = X[t, 64];
        else
            ICC_AP1R_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_AP1R_EL1_S[m] = X[t, 64];
            else
                ICC_AP1R_EL1_NS[m] = X[t, 64];
        else
            ICC_AP1R_EL1[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_AP1R_EL1_S[m] = X[t, 64];
            else
                ICC_AP1R_EL1_NS[m] = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1

The ICC_BPR1_EL1 characteristics are:

Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

Configuration

AArch64 System register ICC_BPR1_EL1 bits [31:0] (ICC_BPR1_EL1_S) are architecturally mapped to AArch32 System register [ICC_BPR1\[31:0\]](#) (ICC_BPR1_S).

AArch64 System register ICC_BPR1_EL1 bits [31:0] (ICC_BPR1_EL1_NS) are architecturally mapped to AArch32 System register [ICC_BPR1\[31:0\]](#) (ICC_BPR1_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_BPR1_EL1 are UNDEFINED.

Virtual accesses to this register update [ICH_VMCR_EL2.VBPR1](#).

Attributes

ICC_BPR1_EL1 is a 64-bit register.

This register has the following instances:

- [ICC_BPR1_EL1](#), when when EL3 is not implemented
- [ICC_BPR1_EL1_S](#), when when EL3 is implemented
- [ICC_BPR1_EL1_NS](#), when when EL3 is implemented

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BinaryPoint																															

Bits [63:3]

Reserved, RES0.

BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see 'Priority grouping' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

The minimum value of the Non-secure copy of this register is the minimum value of [ICC_BPR0_EL1](#) + 1. The minimum value of the Secure copy of this register is the minimum value of [ICC_BPR0_EL1](#).

If EL3 is implemented and [ICC_CTLR_EL3.CBPR_EL1S](#) is 1:

- Accesses to this register from Secure EL2 access the state of [ICC_BPR0_EL1](#).
- Accesses to this register from Secure EL1:
 - When SCR_EL3.EEL2 is 1 and HCR_EL2.IMO is 1, access the state of [ICV_BPR1_EL1](#).
 - Otherwise, access the state of [ICC_BPR0_EL1](#).

If EL3 is implemented and [ICC_CTLR_EL3](#).CBPR_EL1NS is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR_EL2.IMO and SCR_EL3.IRQ:

HCR_EL2.IMO	SCR_EL3.IRQ	Behavior
0b0	0b0	Non-secure EL1 and EL2 reads return ICC_BPR0_EL1 + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0b0	0b1	Non-secure EL1 and EL2 accesses trap to EL3.
0b1	0b0	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0_EL1 + 1 saturated to 0b111. Non-secure EL2 writes are ignored.
0b1	0b1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 accesses trap to EL3.

If EL3 is not implemented and [ICC_CTLR_EL1](#).CBPR is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR_EL2.IMO:

HCR_EL2.IMO	Behavior
0b0	Non-secure EL1 and EL2 reads return ICC_BPR0_EL1 + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0b1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0_EL1 + 1 saturated to 0b111. Non-secure EL2 writes are ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICC_BPR1_EL1

On a reset, the binary point field is UNKNOWN.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_BPR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        X[t, 64] = ICV_BPR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;
    else
        X[t, 64] = ICC_BPR1_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;
    else
        X[t, 64] = ICC_BPR1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;

```

MSR ICC_BPR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        ICV_BPR1_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];
    else
        ICC_BPR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];
    else
        ICC_BPR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

The ICC_CTLR_EL1 characteristics are:

Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configuration

AArch64 System register ICC_CTLR_EL1 bits [31:0] (ICC_CTLR_EL1_S) are architecturally mapped to AArch32 System register [ICC_CTLR\[31:0\]](#) (ICC_CTLR_S).

AArch64 System register ICC_CTLR_EL1 bits [31:0] (ICC_CTLR_EL1_NS) are architecturally mapped to AArch32 System register [ICC_CTLR\[31:0\]](#) (ICC_CTLR_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_CTLR_EL1 are UNDEFINED.

Attributes

ICC_CTLR_EL1 is a 64-bit register.

This register has the following instances:

- ICC_CTLR_EL1, when when EL3 is not implemented
- ICC_CTLR_EL1_S, when when EL3 is implemented
- ICC_CTLR_EL1_NS, when when EL3 is implemented

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0												ExtRange		RSS	RES0	A3V	SEIS	IDbits	PRIbits	RES0	PMHE	RES0		EOImode	CBPR						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

ExtRange	Meaning
0b0	<p>CPU interface does not support INTIDs in the range 1024..8191.</p> <ul style="list-style-type: none"> Behavior is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. <p>Note Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.</p>
0b1	<p>CPU interface supports INTIDs in the range 1024..8191</p> <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation.

If EL3 is implemented, ICC_CTLR_EL1.ExtRange is an alias of [ICC_CTLR_EL3.ExtRange](#).

RSS, bit [18]

Range Selector Support. Possible values are:

RSS	Meaning
0b0	Targeted SGIs with affinity level 0 values of 0 - 15 are supported.
0b1	Targeted SGIs with affinity level 0 values of 0 - 255 are supported.

This bit is read-only.

Bits [17:16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0b0	The CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
0b1	The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

If EL3 is implemented, this bit is an alias of [ICC_CTLR_EL3.A3V](#).

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs:

SEIS	Meaning
0b0	The CPU interface logic does not support local generation of SEIs.
0b1	The CPU interface logic supports local generation of SEIs.

If EL3 is implemented, this bit is an alias of [ICC_CTLR_EL3.SEIS](#).

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
0b000	16 bits.
0b001	24 bits.

All other values are reserved.

If EL3 is implemented, this field is an alias of [ICC_CTLR_EL3.IDbits](#).

PRIBits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

Note

This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of [GICD_CTLR.DS](#).

For physical accesses, this field determines the minimum value of [ICC_BPR0_EL1](#).

If EL3 is implemented, physical accesses return the value from [ICC_CTLR_EL3.PRIBits](#).

If EL3 is not implemented, physical accesses return the value from this field.

Bit [7]

Reserved, RES0.

PMHE, bit [6]

Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:

PMHE	Meaning
0b0	Disables use of ICC_PMR_EL1 as a hint for interrupt distribution.
0b1	Enables use of ICC_PMR_EL1 as a hint for interrupt distribution.

If EL3 is implemented, this bit is an alias of [ICC_CTLR_EL3.PMHE](#). Whether this bit can be written as part of an access to this register depends on the value of [GICD_CTLR.DS](#):

- If [GICD_CTLR.DS](#) == 0, this bit is read-only.
- If [GICD_CTLR.DS](#) == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read/write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

Bits [5:2]

Reserved, RES0.

EOImode, bit [1]

EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode	Meaning
0b0	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
0b1	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

The Secure [ICC_CTLR_EL1](#).EOImode is an alias of [ICC_CTLR_EL3](#).EOImode_EL1S.

The Non-secure [ICC_CTLR_EL1](#).EOImode is an alias of [ICC_CTLR_EL3](#).EOImode_EL1NS

CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:

CBPR	Meaning
0b0	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts.
0b1	ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts.

If EL3 is implemented:

- This bit is an alias of [ICC_CTLR_EL3](#).CBPR_EL1{S,NS} where S or NS corresponds to the current Security state.
- If [GICD_CTLR](#).DS == 0, this bit is read-only.
- If [GICD_CTLR](#).DS == 1, this bit is read/write.

If EL3 is not implemented, this bit is read/write.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICC_CTLR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;

```

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ICC_CTLR_EL3 characteristics are:

Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configuration

This register is present only when FEAT_GICv3 is implemented and EL3 is implemented. Otherwise, direct accesses to ICC_CTLR_EL3 are UNDEFINED.

Attributes

ICC_CTLR_EL3 is a 64-bit register.

Field descriptions

636261605958575655545352	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
RES0																	
RES0	ExtRange	RSSn	DS	RES0	A3V	SEIS	IDbits	PRIBits	RES0	PMHE	RM	EOImode	EL1NS	EOImode	EL1		
313029282726252423222120	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3

Bits [63:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

ExtRange	Meaning
0b0	<p>CPU interface does not support INTIDs in the range 1024..8191.</p> <ul style="list-style-type: none"> Behavior Behaviour is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. <hr/> <p>Note</p> <p>Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.</p> <hr/>
0b1	<p>CPU interface supports INTIDs in the range 1024..8191</p> <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation.

RSS, bit [18]

Range Selector Support.

RSS	Meaning
0b0	Targeted SGIs with affinity level 0 values of 0-15 are supported.
0b1	Targeted SGIs with affinity level 0 values of 0-255 are supported.

This bit is read-only.

nDS, bit [17]

Disable Security not supported. Read-only and writes are ignored.

nDS	Meaning
0b0	The CPU interface logic supports disabling of security.
0b1	The CPU interface logic does not support disabling of security, and requires that security is not disabled.

When a PE implements [FEAT_RME](#) the [and Realm](#) [FEAT_SEL2](#) [Management Extension](#), this field is RAO/WI.

Bit [16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored.

A3V	Meaning
0b0	The CPU interface logic does not support non-zero values of the Aff3 field in SGI generation System registers.
0b1	The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.

If EL3 is present, [ICC_CTLR_EL1](#).A3V is an alias of ICC_CTLR_EL3.A3V

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs:

SEIS	Meaning
0b0	The CPU interface logic does not support generation of SEIs.
0b1	The CPU interface logic supports generation of SEIs.

If EL3 is present, [ICC_CTLR_EL1](#).SEIS is an alias of ICC_CTLR_EL3.SEIS

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported.

IDbits	Meaning
0b000	16 bits.
0b001	24 bits.

All other values are reserved.

If EL3 is present, [ICC_CTLR_EL1](#).IDbits is an alias of ICC_CTLR_EL3.IDbits

PRibits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

Note

This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of [GICD_CTLR.DS](#).

The division between group priority and subpriority is defined in the binary point registers [ICC_BPR0_EL1](#) and [ICC_BPR1_EL1](#).

This field determines the minimum value of ICC_BPR0_EL1.

Bit [7]

Reserved, RES0.

PMHE, bit [6]

Priority Mask Hint Enable.

PMHE	Meaning
0b0	Disables use of the priority mask register as a hint for interrupt distribution.
0b1	Enables use of the priority mask register as a hint for interrupt distribution.

Software must write [ICC_PMR_EL1](#) to 0xFF before clearing this field to 0.

- An implementation might choose to make this field RAO/WI if priority-based routing is always used
- An implementation might choose to make this field RAZ/WI if priority-based routing is never used

If EL3 is present, [ICC_CTLR_EL1](#).PMHE is an alias of ICC_CTLR_EL3.PMHE.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

RM, bit [5]

Routing Modifier. This bit controls whether EL3 can acknowledge, or observe as the Highest Priority Pending Interrupt, Secure Group 0 and Non-secure Group 1 interrupts.

RM	Meaning
0b0	Secure Group 0 and Non-secure Group 1 interrupts can be acknowledged and observed as the highest priority interrupt at EL3.
0b1	Secure Group 0 and Non-secure Group 1 interrupts cannot be acknowledged and observed as the highest priority interrupt at EL3. Secure Group 0 interrupts return a special INTID value of 1020. This affects accesses to ICC_IAR0_EL1 and ICC_HPPIR0_EL1 . Non-secure Group 1 interrupts return a special INTID value of 1021. This affects accesses to ICC_IAR1_EL1 and ICC_HPPIR1_EL1 .

Note

The Routing Modifier bit is supported in AArch64 only. In systems without EL3 the behavior is as if the value is 0. Software must ensure this bit is 0 when the Secure copy of [ICC_SRE_EL1](#).SRE is 1, otherwise system behavior is UNPREDICTABLE. In systems without EL3 or where the Secure copy of [ICC_SRE_EL1](#).SRE is RAO/WI, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL1NS, bit [4]

EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL1NS	Meaning
0b0	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
0b1	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

If EL3 is present, [ICC_CTLR_EL1](#)(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL1S, bit [3]

EOI mode for interrupts handled at Secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL1S	Meaning
0b0	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
0b1	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

If EL3 is present, [ICC_CTLR_EL1](#)(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EOImode_EL3, bit [2]

EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL3	Meaning
0b0	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are UNPREDICTABLE.
0b1	ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR_EL1NS, bit [1]

Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.

CBPR_EL1NS	Meaning
0b0	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts.
0b1	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to GICC_BPR and ICC_BPR1_EL1 access the state of ICC_BPR0_EL1 .

If EL3 is present, [ICC_CTLR_EL1\(NS\)](#).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR_EL1S, bit [0]

Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1 and EL2.

CBPR_EL1S	Meaning
0b0	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts.
0b1	ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to ICC_BPR1_EL1 access the state of ICC_BPR0_EL1 .

If EL3 is present, [ICC_CTLR_EL1\(S\)](#).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICC_CTLR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_CTLR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_CTLR_EL3;

```

MSR ICC_CTLR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_CTLR_EL3 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_IGRPEN0_EL1, Interrupt Controller Interrupt Group 0 Enable register

The ICC_IGRPEN0_EL1 characteristics are:

Purpose

Controls whether Group 0 interrupts are enabled or not.

Configuration

AArch64 System register ICC_IGRPEN0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ICC_IGRPEN0\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_IGRPEN0_EL1 are UNDEFINED.

Attributes

ICC_IGRPEN0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															Enable
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:1]

Reserved, RES0.

Enable, bit [0]

Enables Group 0 interrupts.

Enable	Meaning
0b0	Group 0 interrupts are disabled.
0b1	Group 0 interrupts are enabled.

Virtual accesses to this register update [ICH_VMCR_EL2.VENG0](#).

If the highest priority pending interrupt for that PE is a Group 0 interrupt using 1 of N model, then the interrupt will be targeted to another PE as a result of the Enable bit changing from 1 to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_IGRPEN0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_IGRPEN0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IGRPEN0_EL1;
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ICC_IGRPEN0_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ICC_IGRPEN0_EL1;
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ICC_IGRPEN0_EL1;

```

MSR ICC_IGRPEN0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_IGRPEN0_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ICC_IGRPEN0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ICC_IGRPEN0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ICC_IGRPEN0_EL1 = X[t, 64];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

- The Non-secure [ICC_IGRPEN1_EL1](#).Enable bit is a read/write alias of the [ICC_IGRPEN1_EL3](#).EnableGrp1NS bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_IGRPEN1_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_IGRPEN1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IGRPEN1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;

```

MSR ICC_IGRPEN1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
& HFGWTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_IGRPEN1_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_SRE_EL1, Interrupt Controller System Register Enable register (EL1)

The ICC_SRE_EL1 characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL1.

Configuration

AArch64 System register ICC_SRE_EL1 bits [31:0] (ICC_SRE_EL1_S) are architecturally mapped to AArch32 System register [ICC_SRE\[31:0\]](#) (ICC_SRE_S).

AArch64 System register ICC_SRE_EL1 bits [31:0] (ICC_SRE_EL1_NS) are architecturally mapped to AArch32 System register [ICC_SRE\[31:0\]](#) (ICC_SRE_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_SRE_EL1 are UNDEFINED.

Attributes

ICC_SRE_EL1 is a 64-bit register.

This register has the following instances:

- ICC_SRE_EL1, when when EL3 is not implemented
- ICC_SRE_EL1_S, when when EL3 is implemented
- ICC_SRE_EL1_NS, when when EL3 is implemented

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
RES0																															
DIBDFBSRE																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

If [GICD_CTLR.DS](#) == 1 and EL2 is implemented, this field is a read-only alias of [ICC_SRE_EL2.DIB](#).

In systems that do not support IRQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0b0	FIQ bypass enabled.
0b1	FIQ bypass disabled.

If EL3 is implemented and [GICD_CTLR.DS](#) == 0, this field is a read-only alias of [ICC_SRE_EL3.DFB](#).

If EL3 is implemented and [GICD_CTLR.DS](#) == 1, and EL2 is not implemented, this field is a read/write alias of [ICC_SRE_EL3.DFB](#).

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of [ICC_SRE_EL2.DFB](#).

If [GICD_CTLR.DS](#) == 1 and EL2 is implemented, this field is a read-only alias of [ICC_SRE_EL2.DFB](#).

In systems that do not support FIQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

SRE, bit [0]

System Register Enable.

SRE	Meaning
0b0	The memory-mapped interface must be used. Access at EL1 to any ICC_* System register other than ICC_SRE_EL1 is trapped to EL1.
0b1	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and [ICC_SRE_EL3.SRE](#)==0 the Secure copy of this bit is RAZ/WI. If [ICC_SRE_EL3.SRE](#) is changed from zero to one, the Secure copy of this bit becomes UNKNOWN.

If EL2 is implemented and [ICC_SRE_EL2.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI. If [ICC_SRE_EL2.SRE](#) is changed from zero to one, the Non-secure copy of this bit becomes UNKNOWN.

If EL3 is implemented and [ICC_SRE_EL3.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI. If [ICC_SRE_EL3.SRE](#) is changed from zero to one, the Non-secure copy of this bit becomes UNKNOWN.

If Realm Management Extension is implemented, this field is RAO/WI.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI. The following options are supported:

- The Non-secure copy of [ICC_SRE_EL1.SRE](#) can be RAO/WI if [ICC_SRE_EL2.SRE](#) is also RAO/WI. This means all Non-secure software, including VMs using only virtual interrupts, must access the GIC using System registers.
- The Secure copy of [ICC_SRE_EL1.SRE](#) can be RAO/WI if [ICC_SRE_EL3.SRE](#) and [ICC_SRE_EL2.SRE](#) are also RAO/WI. This means that all Secure software must access the GIC using System registers and all Non-secure accesses to registers for physical interrupts must use System registers.

Note

A VM using only virtual interrupts might still use memory-mapped access if the Non-secure copy of [ICC_SRE_EL1](#).SRE is not RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_SRE_EL1

Execution with [ICC_SRE_EL1](#).SRE set to 0 might make some System registers UNKNOWN.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_SRE_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_SRE_EL1_S;
        else
            X[t, 64] = ICC_SRE_EL1_NS;
    else
        X[t, 64] = ICC_SRE_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_SRE_EL1_S;
        else
            X[t, 64] = ICC_SRE_EL1_NS;
    else
        X[t, 64] = ICC_SRE_EL1;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        X[t, 64] = ICC_SRE_EL1_S;
    else
        X[t, 64] = ICC_SRE_EL1_NS;

```

MSR ICC_SRE_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t, 64];
        else
            ICC_SRE_EL1_NS = X[t, 64];
    else
        ICC_SRE_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t, 64];
        else
            ICC_SRE_EL1_NS = X[t, 64];
    else
        ICC_SRE_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_SRE_EL1_S = X[t, 64];
    else
        ICC_SRE_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

The ICV_CTLR_EL1 characteristics are:

Purpose

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configuration

AArch64 System register ICV_CTLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ICV_CTLR\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_CTLR_EL1 are UNDEFINED.

Attributes

ICV_CTLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0												ExtRange	RSS	RES0	A3V	SEIS	IDbits	PRIbits	RES0						EOImode	CBPR					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

ExtRange	Meaning
0b0	CPU interface does not support INTIDs in the range 1024..8191. <ul style="list-style-type: none">BehaviorBehaviour is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. <div>Note<p>Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.</p></div>
0b1	CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none">All INTIDs in the range 1024..8191 are treated as requiring deactivation.

ICV_CTLR_EL1.ExtRange is an alias of [ICC_CTLR_EL1](#).ExtRange.

RSS, bit [18]

Range Selector Support. Possible values are:

RSS	Meaning
0b0	Targeted SGIs with affinity level 0 values of 0 - 15 are supported.
0b1	Targeted SGIs with affinity level 0 values of 0 - 255 are supported.

This bit is read-only.

Bits [17:16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0b0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
0b1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs:

SEIS	Meaning
0b0	The virtual CPU interface logic does not support local generation of SEIs.
0b1	The virtual CPU interface logic supports local generation of SEIs.

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported:

IDbits	Meaning
0b000	16 bits.
0b001	24 bits.

All other values are reserved.

PRIbits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation must implement at least 32 levels of physical priority (5 priority bits).

Note

This field always returns the number of priority bits implemented.

The division between group priority and subpriority is defined in the binary point registers [ICV_BPR0_EL1](#) and [ICV_BPR1_EL1](#).

Bits [7:2]

Reserved, RES0.

EOImode, bit [1]

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

EOImode	Meaning
0b0	ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are UNPREDICTABLE.
0b1	ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR_EL1 provides interrupt deactivation functionality.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:

CBPR	Meaning
0b0	ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.
0b1	Non-secure reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 0b111. Non-secure writes to ICV_BPR1_EL1 are ignored. Secure reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 . Secure writes of ICV_BPR1_EL1 modify ICV_BPR0_EL1 .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICV_CTLR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;

```

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICV_IGRPEN0_EL1, Interrupt Controller Virtual Interrupt Group 0 Enable register

The ICV_IGRPEN0_EL1 characteristics are:

Purpose

Controls whether virtual Group 0 interrupts are enabled or not.

Configuration

AArch64 System register ICV_IGRPEN0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ICV_IGRPEN0\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_IGRPEN0_EL1 are UNDEFINED.

Attributes

ICV_IGRPEN0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															Enable
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:1]

Reserved, RES0.

Enable, bit [0]

Enables virtual Group 0 interrupts.

Enable	Meaning
0b0	Virtual Group 0 interrupts are disabled.
0b1	Virtual Group 0 interrupts are enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICV_IGRPEN0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ICC_IGRPEN0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IGRPEN0_EL1;
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_IGRPEN0_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_IGRPEN0_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_IGRPEN0_EL1;

```

MSR ICC_IGRPEN0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        ICV_IGRPEN0_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ICC_IGRPEN0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.FIQ == '1' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                ICC_IGRPEN0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            ICC_IGRPEN0_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IGRPEN1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_IGRPEN1_EL1_S;
        else
            X[t, 64] = ICC_IGRPEN1_EL1_NS;

```

MSR ICC_IGRPEN1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
& HFGWTR_EL2.ICC_IGRPENn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_IGRPEN1_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_IGRPEN1_EL1_S = X[t, 64];
        else
            ICC_IGRPEN1_EL1_NS = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

The ID_AA64DFR0_EL1 characteristics are:

Purpose

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

The external register [EDDFR](#) gives information from this register.

Attributes

ID_AA64DFR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
HPMN0				ExtTrcBuffRES0				BRBE				MTPMU				TraceBuffer				TraceFilt				DoubleLock				PMSVer			
CTX_CMPs				SEBEPRES0				WRPs				PMSSRES0				BRPs				PMUVer				TraceVer				DebugVer			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

HPMN0, bits [63:60]

Zero PMU event counters for a Guest operating system. Defined values are:

HPMN0	Meaning
0b0000	Setting MDCR_EL2 .HPMN to zero has CONSTRAINED UNPREDICTABLE behavior.
0b0001	Setting MDCR_EL2 .HPMN to zero has defined behavior.

All other values are reserved.

If FEAT_PMUv3 is not implemented, FEAT_FGT is not implemented, or EL2 is not implemented, the only permitted value is 0b0000.

FEAT_HPMN0 implements the functionality identified by the value 0b0001.

From Armv8.8, in an implementation that includes FEAT_PMUv3, FEAT_FGT, and EL2, the value 0b0000 is not permitted.

ExtTrcBuff, bits [59:56]

Trace Buffer External Mode Extension. Defined values are:

Reserved, RES0.

ExtTrcBuff	Meaning
0b0000	Trace Buffer External Mode not implemented.
0b0001	Trace Buffer External Mode implemented.

All other values are reserved.

FEAT_TRBE_EXT implements the functionality identified by the value 0b0001.

BRBE, bits [55:52]

Branch Record Buffer Extension. Defined values are:

BRBE	Meaning
0b0000	Branch Record Buffer Extension not implemented.
0b0001	Branch Record Buffer Extension implemented.
0b0010	As 0b0001, and adds support for branch recording at EL3.

All other values are reserved.

FEAT_BRBE implements the functionality identified by the value 0b0001.

FEAT_BRBEv1p1 implements the functionality identified by the value 0b0010.

From Armv9.3, if FEAT_BRBE is implemented, the value 0b0001 is not permitted.

MTPMU, bits [51:48]

Multi-threaded PMU extension. Defined values are:

MTPMU	Meaning
0b0000	FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, it is IMPLEMENTATION DEFINED whether PMEVTYPER<n>_EL0.MT and PMEVTYPER<n>.MT are read/write or RES0.
0b0001	FEAT_MTPMU and FEAT_PMUv3 implemented. PMEVTYPER<n>_EL0.MT and PMEVTYPER<n>.MT are read/write. When FEAT_MTPMU is disabled, the Effective values of PMEVTYPER<n>_EL0.MT and PMEVTYPER<n>.MT are 0.
0b1111	FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, PMEVTYPER<n>_EL0.MT and PMEVTYPER<n>.MT are RES0.

All other values are reserved.

FEAT_MTPMU implements the functionality identified by the value 0b0001.

From Armv8.6, in an implementation that includes FEAT_PMUv3, the value 0b0000 is not permitted.

In an implementation that does not include FEAT_PMUv3, the value 0b0001 is not permitted.

TraceBuffer, bits [47:44]

Trace Buffer Extension. Defined values are:

TraceBuffer	Meaning
0b0000	Trace Buffer Extension not implemented.
0b0001	Trace Buffer Extension implemented.

All other values are reserved.

FEAT_TRBE implements the functionality identified by the value 0b0001.

In any Armv9 implementation, if FEAT_ETE is implemented, the value 0b0000 is not permitted.

TraceFilt, bits [43:40]

Armv8.4 Self-hosted Trace Extension version. Defined values are:

TraceFilt	Meaning
0b0000	Armv8.4 Self-hosted Trace Extension not implemented.
0b0001	Armv8.4 Self-hosted Trace Extension implemented.

All other values are reserved.

FEAT_TRF implements the functionality identified by the value 0b0001.

From Armv8.4, if an Embedded Trace Macrocell Architecture trace unit is implemented, the value 0b0000 is not permitted.

DoubleLock, bits [39:36]

OS Double Lock implemented. Defined values are:

DoubleLock	Meaning
0b0000	OS Double Lock implemented. OSDLR_EL1 is RW.
0b1111	OS Double Lock not implemented. OSDLR_EL1 is RAZ/WI.

All other values are reserved.

FEAT_DoubleLock implements the functionality identified by the value 0b0000.

In Armv8.0, the only permitted value is 0b0000.

If FEAT_Debugv8p2 is implemented and FEAT_DoPD is not implemented, the permitted values are 0b0000 and 0b1111.

If FEAT_DoPD is implemented, the only permitted value is 0b1111.

PMSVer, bits [35:32]

Statistical Profiling Extension version. Defined values are:

PMSVer	Meaning
0b0000	Statistical Profiling Extension not implemented.
0b0001	Statistical Profiling Extension implemented.
0b0010	As 0b0001, and adds: <ul style="list-style-type: none"> Support for the Events packet Alignment flag. If FEAT_SVE is implemented, support for the Scalable Vector extensions to Statistical Profiling.
0b0011	As 0b0010, and adds: <ul style="list-style-type: none"> Discard mode. Extended event filtering, including the PMSNEVER_EL1 System register. Support for the OPTIONAL previous branch target Address packet. If FEAT_PMUv3 is implemented, controls to freeze the PMU event counters after an SPE buffer management event occurs. If FEAT_PMUv3 is implemented, the SAMPLE_FEED_BR, SAMPLE_FEED_EVENT, SAMPLE_FEED_LAT, SAMPLE_FEED_LD, SAMPLE_FEED_OP, and SAMPLE_FEED_ST PMU events.
0b0100	As 0b0011, and adds: <ul style="list-style-type: none"> If FEAT_MOPS is implemented, Operation Type packet encodings for Memory Copy and Set operations. If FEAT_MTE is implemented, Operation Type packet encodings for loads and stores of Allocation Tags.
0b0101	As 0b0100, and adds: <ul style="list-style-type: none"> Support for the Events packet Level 2 Data cache access, Level 2 Data cache miss, Cached data modified, Recently fetched cache line, and Cache snoop flags. Support for Data Source filtering.

All other values are reserved.

FEAT_SPE implements the functionality identified by the value 0b0001.

FEAT_SPEv1p1 implements the functionality identified by the value 0b0010.

FEAT_SPEv1p2 implements the functionality identified by the value 0b0011.

FEAT_SPEv1p3 implements the functionality identified by the value 0b0100.

FEAT_SPEv1p4 implements the functionality identified by the value 0b0101.

From Armv8.5, if FEAT_SPE is implemented, the value 0b0001 is not permitted.

From Armv8.7, if FEAT_SPE is implemented, the value 0b0010 is not permitted.

From Armv8.8, if FEAT_SPE is implemented, the value 0b0011 is not permitted.

From Armv8.9, if FEAT_SPE is implemented, the value 0b0100 is not permitted.

CTX_CMPs, bits [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

If FEAT_Debugv8p9 is implemented and 16 or more breakpoints that are context-aware are implemented, this field reads as 0b1111.

SEBEP, bits [27:24]

Synchronous-exception-based event profiling. Defined values are:

Reserved, RES0.

SEBEP	Meaning
0b0000	Synchronous-exception-based event profiling not implemented.
0b0001	Synchronous-exception-based event profiling implemented.

All other values are reserved.

FEAT_SEBEP implements the functionality identified by the value 0b0001.

WRPs, bits [23:20]

Number of watchpoints, minus 1.

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

If FEAT_Debugv8p9 is implemented and 16 or more watchpoints are implemented, this field reads as 0b1111.

The value of 0b0000 is reserved.

PMSS, bits [19:16]

PMU Snapshot extension. Defined values are:

Reserved, RES0.

PMSS	Meaning
0b0000	PMU snapshot extension not implemented.
0b0001	PMU snapshot extension implemented.

All other values are reserved.

FEAT_PMUv3_SS implements the functionality identified by the value 0b0001.

BRPs, bits [15:12]

Number of breakpoints, minus 1.

Number of breakpoints, minus 1. The value of 0b0000 is reserved.

If FEAT_Debugv8p9 is implemented and 16 or more breakpoints are implemented, this field reads as 0b1111.

The value of 0b0000 is reserved.

PMUVer, bits [11:8]

Performance Monitors Extension version.

This field does not follow the standard ID scheme, but uses the alternative ID scheme described in 'Alternative ID scheme used for the Performance Monitors Extension version'

Defined values are:

PMUVer	Meaning
0b0000	Performance Monitors Extension not implemented.
0b0001	Performance Monitors Extension, PMUv3 implemented.
0b0100	PMUv3 for Armv8.1. As 0b0001, and adds support for: <ul style="list-style-type: none"> Extended 16-bit PMEVTYPER<n>_EL0.evtCount field. If EL2 is implemented, the MDCR_EL2.HPMD control.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds support for the PMMIR_EL1 register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the MDCR_EL2.HCCD control. If EL3 is implemented, the MDCR_EL3.SCCD control.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds support for: <ul style="list-style-type: none"> The PMCR_EL0.FZO and, if EL2 is implemented, MDCR_EL2.HPMFZO controls. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls.
0b1000	PMUv3 for Armv8.8. As 0b0111, and: <ul style="list-style-type: none"> Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. Removes the CONSTRAINED UNPREDICTABLE behaviors if a reserved or unimplemented PMU event number is selected.
0b1001	PMUv3 for Armv8.9. As 0b1000, and: <ul style="list-style-type: none"> Updates the definitions of existing PMU events. Adds support for the PMUSERENR_EL0.UEN control and the PMUACR_EL1 register. Adds support for the EDECR.PME control.
0b1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. Arm does not recommend this value for new implementations.

All other values are reserved.

FEAT_PMUv3 implements the functionality identified by the value 0b0001.

FEAT_PMUv3p1 implements the functionality identified by the value 0b0100.

FEAT_PMUv3p4 implements the functionality identified by the value 0b0101.

FEAT_PMUv3p5 implements the functionality identified by the value 0b0110.

FEAT_PMUv3p7 implements the functionality identified by the value 0b0111.

FEAT_PMUv3p8 implements the functionality identified by the value 0b1000.

FEAT_PMUv3p9 implements the functionality identified by the value 0b1001.

From Armv8.1, if FEAT_PMUv3 is implemented, the value 0b0001 is not permitted.

From Armv8.4, if FEAT_PMUv3 is implemented, the value 0b0100 is not permitted.

From Armv8.5, if FEAT_PMUv3 is implemented, the value 0b0101 is not permitted.

From Armv8.7, if FEAT_PMUv3 is implemented, the value 0b0110 is not permitted.

From Armv8.8, if FEAT_PMUv3 is implemented, the value 0b0111 is not permitted.

From Armv8.9, if FEAT_PMUv3 is implemented, the value 0b1000 is not permitted.

TraceVer, bits [7:4]

Trace support. Indicates whether System register interface to a trace unit is implemented. Defined values are:

TraceVer	Meaning
0b0000	Trace unit System registers not implemented.
0b0001	Trace unit System registers implemented.

All other values are reserved.

When trace unit System registers are implemented, see [TRCIDR1](#) for tracing capabilities of the trace unit.

DebugVer, bits [3:0]

Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are:

DebugVer	Meaning
0b0110	Armv8 debug architecture.
0b0111	Armv8 debug architecture with Virtualization Host Extensions.
0b1000	Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.
0b1011	Armv8.9 debug architecture, FEAT_Debugv8p9.

All other values are reserved.

FEAT_VHE adds the functionality identified by the value 0b0111.

FEAT_Debugv8p2 adds the functionality identified by the value 0b1000.

FEAT_Debugv8p4 adds the functionality identified by the value 0b1001.

FEAT_Debugv8p8 adds the functionality identified by the value 0b1010.

FEAT_Debugv8p9 adds the functionality identified by the value 0b1011.

From Armv8.1, when FEAT_VHE is implemented the value 0b0110 is not permitted.

From Armv8.2, the values 0b0110 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 is not permitted.

From Armv8.9, the value 0b1010 is not permitted.

Accessing ID_AA64DFR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64DFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_AA64DFR0_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = ID_AA64DFR0_EL1;
    elseif PSTATE.EL == EL3 then
        X[t, 64] = ID_AA64DFR0_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

The ID_AA64DFR1_EL1 characteristics are:

Purpose

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Provides Reserved for future expansion of top level information about the debug system in AArch64 state.

Configuration

There are no configuration notes.

Attributes

ID_AA64DFR1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ABL_CMPs, Bits bits [63:560]

When FEAT_ABLE is implemented:

Number of breakpoints that support address linking, minus 1.

The values 0x40 to 0xFF are reserved.

The value of this field is never greater than either ID_AA64DFR1_EL1.WRPs or ID_AA64DFR1_EL1.BRPs.

Otherwise:

Reserved, RES0.

Bits [55:52]

Reserved, RES0.

EBEP, bits [51:48]

Exception-based event profiling. Defined values are:

EBEP	Meaning
0b0000	Exception-based event profiling not implemented.
0b0001	Exception-based event profiling implemented.

All other values are reserved.

FEAT_EBEP implements the functionality identified by the value 0b0001.

ITE, bits [47:44]

Instrumentation Trace Extension. Defined values are:

ITE	Meaning
0b0000	Instrumentation Trace Extension not implemented.
0b0001	Instrumentation Trace Extension implemented.

All other values are reserved.

FEAT_ITE implements the functionality identified by the value 0b0001.

If FEAT_ITE is not implemented, then the only permitted value is 0b0000.

From Armv9.4, when FEAT_ITE is implemented, the value 0b0000 is not permitted.

ABLE, bits [43:40]

Address Breakpoint Linking Extension. Defined values are:

ABLE	Meaning
0b0000	Address Breakpoint Linking Extension not implemented.
0b0001	Address Breakpoint Linking Extension implemented.

All other values are reserved.

FEAT_ABLE implements the functionality identified by the value 0b0001.

If FEAT_ABLE is not implemented, then the only permitted value is 0b0000.

From Armv9.4, when FEAT_ABLE is implemented, the value 0b0000 is not permitted.

PMICNTR, bits [39:36]

PMU fixed-function instruction counter. Defined values are:

PMICNTR	Meaning
0b0000	PMU fixed-function instruction counter not implemented.
0b0001	PMU fixed-function instruction counter implemented.

All other values are reserved.

FEAT_PMUv3_ICNTR implements the functionality identified by the value 0b0001.

If FEAT_PMUv3 is not implemented, then the only permitted value is 0b0000.

SPMU, bits [35:32]

System PMU extension.

SPMU	Meaning
0b0000	System PMU extension not implemented.
0b0001	System PMU extension implemented.

All other values are reserved.

FEAT_SPMU implements the functionality identified by the value 0b0001.

CTX_CMPs, bits [31:24]

When FEAT_Debugv8p9 is implemented and ID_AA64DFR0_EL1.CTX_CMPs == 0b1111:

Number of breakpoints that are context-aware, minus 1.

The value 0x00 means 16 breakpoints that are context-aware are implemented. Otherwise, this field is the number of breakpoints that are context-aware, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

The value of this field is never greater than ID_AA64DFR1_EL1.BRPs.

Otherwise:

Reserved, RES0.

WRPs, bits [23:16]

When FEAT_Debugv8p9 is implemented and ID_AA64DFR0_EL1.WRPs == 0b1111:

Number of watchpoints, minus 1.

The value 0x00 means 16 watchpoints are implemented. Otherwise, this field is the number of watchpoints, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

Otherwise:

Reserved, RES0.

BRPs, bits [15:8]

When FEAT_Debugv8p9 is implemented and ID_AA64DFR0_EL1.BRPs == 0b1111:

Number of breakpoints, minus 1.

The value 0x00 means 16 breakpoints are implemented. Otherwise, this field is the number of breakpoints, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

Otherwise:

Reserved, RES0.

SYSPMUID, bits [7:0]

When FEAT_SPMU is implemented:

System PMU ID. Indicates the largest value of [SPMSELR_ELO](#).SYSPMUSEL.

Since System PMUs might not be contiguously accessible, this field does not necessarily indicate the total number of accessible System PMUs.

The values 0x20 to 0xFF are reserved.

Otherwise:

Reserved, RES0.

Accessing ID_AA64DFR1_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b001

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_AA64DFR1_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ID_AA64DFR1_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ID_AA64DFR1_EL1;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

The ID_AA64ISAR0_EL1 characteristics are:

Purpose

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64ISAR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RNDR					TLB				TS				FHM			DP				SM4			SM3				SHA3				
RDM					TME				Atomic				CRC32			SHA2				SHA1			AES				RES0				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RNDR, bits [63:60]

Indicates support for Random Number instructions in AArch64 state.

When FEAT_RNG_TRAP is implemented, the value returned by a direct read of ID_AA64ISAR0_EL1.RNDR is further controlled by the value of [SCR_EL3.TRNDR](#).

Defined values are:

RNDR	Meaning
0b0000	No Random Number instructions are implemented.
0b0001	RNDR and RNDRRS registers are implemented.

All other values are reserved.

FEAT_RNG implements the functionality identified by the value 0b0001.

From Armv8.5, the permitted values are 0b0000 and 0b0001.

TLB, bits [59:56]

Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are:

TLB	Meaning
0b0000	Outer Shareable and TLB range maintenance instructions are not implemented.
0b0001	Outer Shareable TLB maintenance instructions are implemented.
0b0010	Outer Shareable and TLB range maintenance instructions are implemented.

All other values are reserved.

FEAT_TLBIOS implements the functionality identified by the values 0b0001 and 0b0010.

FEAT_TLBIRANGE implements the functionality identified by the value 0b0010.

From Armv8.4, the only permitted value is 0b0010.

TS, bits [55:52]

Indicates support for flag manipulation instructions. Defined values are:

TS	Meaning
0b0000	No flag manipulation instructions are implemented.
0b0001	CFINV, RMIF, SETF16, and SETF8 instructions are implemented.
0b0010	CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.

All other values are reserved.

FEAT_FlagM implements the functionality identified by the value 0b0001.

FEAT_FlagM2 implements the functionality identified by the value 0b0010.

In Armv8.2, the permitted values are 0b0000 and 0b0001.

In Armv8.4, the only permitted value is 0b0001.

From Armv8.5, the only permitted value is 0b0010.

FHM, bits [51:48]

Indicates support for FMLAL and FMLSL instructions. Defined values are:

FHM	Meaning
0b0000	FMLAL and FMLSL instructions are not implemented.
0b0001	FMLAL and FMLSL instructions are implemented.

All other values are reserved.

FEAT_FHM implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

DP, bits [47:44]

Indicates support for Dot Product instructions in AArch64 state. Defined values are:

DP	Meaning
0b0000	No Dot Product instructions implemented.
0b0001	UDOT and SDOT instructions implemented.

All other values are reserved.

FEAT_DotProd implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

SM4, bits [43:40]

Indicates support for SM4 instructions in AArch64 state. Defined values are:

SM4	Meaning
0b0000	No SM4 instructions implemented.
0b0001	SM4E and SM4EKEY instructions implemented.

All other values are reserved.

If FEAT_SM4 is not implemented, the value 0b0001 is reserved.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

This field must have the same value as ID_AA64ISAR0_EL1.SM3.

SM3, bits [39:36]

Indicates support for SM3 instructions in AArch64 state. Defined values are:

SM3	Meaning
0b0000	No SM3 instructions implemented.
0b0001	SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 instructions implemented.

All other values are reserved.

If FEAT_SM3 is not implemented, the value 0b0001 is reserved.

FEAT_SM3 implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

This field must have the same value as ID_AA64ISAR0_EL1.SM4.

SHA3, bits [35:32]

Indicates support for SHA3 instructions in AArch64 state. Defined values are:

SHA3	Meaning
0b0000	No SHA3 instructions implemented.
0b0001	EOR3, RAX1, XAR, and BCAX instructions implemented.

All other values are reserved.

If FEAT_SHA3 is not implemented, the value 0b0001 is reserved.

FEAT_SHA3 implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

If the value of ID_AA64ISAR0_EL1.SHA1 is 0b0000, this field must have the value 0b0000.

If the value of this field is 0b0001, ID_AA64ISAR0_EL1.SHA2 must have the value 0b0010.

RDM, bits [31:28]

Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are:

RDM	Meaning
0b0000	No RDMA instructions implemented.
0b0001	SQRDMLAH and SQRDMLSH instructions implemented.

All other values are reserved.

FEAT_RDM implements the functionality identified by the value 0b0001.

From Armv8.1, the only permitted value is 0b0001.

TME, bits [27:24]

Indicates support for TME instructions. Defined values are:

TME	Meaning
0b0000	TME instructions are not implemented.
0b0001	TCANCEL, TCOMMIT, TSTART, and TTEST instructions are implemented.

All other values are reserved.

Accessing this field has the following behavior:

- Access is **RAZ/WI** if all of the following are true:
 - PSTATE.EL IN {EL2, EL1}
 - SCR_EL3.TME == 0
- Access is **RAZ/WI** if all of the following are true:
 - PSTATE.EL == EL1
 - EL2Enabled()
 - HCR_EL2.TME == 0
- Otherwise, access to this field is **RO**.

Atomic, bits [23:20]

Indicates support for Atomic instructions in AArch64 state. Defined values are:

Atomic	Meaning	Applies when
0b0000	No Atomic instructions implemented.	
0b0010	LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	
0b0011	As for 0b0010, plus 128-bit instructions LDCLRP, LDSETP and SWPP.	When FEAT_LSE128 is implemented

All other values are reserved.

FEAT_LSE implements the functionality identified by the value 0b0010.

FEAT_LSE128From implementsArmv8.1, the functionalityonly identifiedpermitted byvalue the valueis 0b00110b0010.

From Armv8.1, the value 0b0000 is not permitted.

CRC32, bits [19:16]

Indicates support for CRC32 instructions in AArch64 state. Defined values are:

CRC32	Meaning
0b0000	No CRC32 instructions implemented.
0b0001	CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.

All other values are reserved.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.1, the only permitted value is 0b0001.

SHA2, bits [15:12]

Indicates support for SHA2 instructions in AArch64 state. Defined values are:

SHA2	Meaning
0b0000	No SHA2 instructions implemented.
0b0001	Implements instructions: SHA256H, SHA256H2, SHA256SU0, and SHA256SU1.
0b0010	Implements instructions: <ul style="list-style-type: none"> SHA256H, SHA256H2, SHA256SU0, and SHA256SU1. SHA512H, SHA512H2, SHA512SU0, and SHA512SU1.

All other values are reserved.

FEAT_SHA256 implements the functionality identified by the value 0b0001.

FEAT_SHA512 implements the functionality identified by the value 0b0010.

In Armv8, the permitted values are 0b0000 and 0b0001.

From Armv8.2, the permitted values are 0b0000, 0b0001, and 0b0010.

If the value of ID_AA64ISAR0_EL1.SHA1 is 0b0000, this field must have the value 0b0000.

If the value of this field is 0b0010, ID_AA64ISAR0_EL1.SHA3 must have the value 0b0001.

SHA1, bits [11:8]

Indicates support for SHA1 instructions in AArch64 state. Defined values are:

SHA1	Meaning
0b0000	No SHA1 instructions implemented.
0b0001	SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented.

All other values are reserved.

FEAT_SHA1 implements the functionality identified by the value 0b0001.

From Armv8, the permitted values are 0b0000 and 0b0001.

If the value of ID_AA64ISAR0_EL1.SHA2 is 0b0000, this field must have the value 0b0000.

AES, bits [7:4]

Indicates support for AES instructions in AArch64 state. Defined values are:

AES	Meaning
0b0000	No AES instructions implemented.
0b0001	AESE, AESD, AESMC, and AESIMC instructions implemented.
0b0010	As for 0b0001, plus PMULL and PMULL2 instructions operating on 64-bit source data elements quantities .

FEAT_AES implements the functionality identified by the value 0b0001.

FEAT_PMULL implements the functionality identified by the value 0b0010.

All other values are reserved.

From Armv8, the permitted values are 0b0000 and 0b0010.

Bits [3:0]

Reserved, RES0.

MRS <Xt>, ID_AA64ISAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR0_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

The ID_AA64ISAR1_EL1 characteristics are:

Purpose

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64ISAR1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
LS64				XS				I8MM				DGH				BF16				SPECRES				SB				FRINTTS			
GPI				GPA				LRCPC				FCMA				JSCVT				API				APA				DPB			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LS64, bits [63:60]

Indicates support for LD64B and ST64B* instructions, and the [ACCDATA_EL1](#) register. Defined values of this field are:

LS64	Meaning
0b0000	The LD64B, ST64B, ST64BV, and ST64BV0 instructions, the ACCDATA_EL1 register, and associated traps are not supported.
0b0001	The LD64B and ST64B instructions are supported.
0b0010	The LD64B, ST64B, and ST64BV instructions, and their associated traps are supported.
0b0011	The LD64B, ST64B, ST64BV, and ST64BV0 instructions, the ACCDATA_EL1 register, and their associated traps are supported.

All other values are reserved.

FEAT_LS64 implements the functionality identified by 0b0001.

FEAT_LS64_V implements the functionality identified by 0b0010.

FEAT_LS64_ACCDATA implements the functionality identified by 0b0011.

From Armv8.7, the permitted values are 0b0000, 0b0001, 0b0010, and 0b0011.

XS, bits [59:56]

Indicates support for the XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the [HCRX_EL2](#).{FGTnXS, FnXS} fields in AArch64 state. Defined values are:

XS	Meaning
0b0000	The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2 .{FGTnXS, FnXS} fields are not supported.
0b0001	The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2 .{FGTnXS, FnXS} fields are supported.

All other values are reserved.

FEAT_XS implements the functionality identified by 0b0001.

From Armv8.7, the only permitted value is 0b0001.

I8MM, bits [55:52]

Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values are:

I8MM	Meaning
0b0000	Int8 matrix multiplication instructions are not implemented.
0b0001	SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.

All other values are reserved.

FEAT_I8MM implements the functionality identified by 0b0001.

When Advanced SIMD and SVE are both implemented, this field must return the same value as [ID_AA64ZFR0_EL1](#).I8MM.

From Armv8.6, the only permitted value is 0b0001.

DGH, bits [51:48]

Indicates support for the Data Gathering Hint instruction. Defined values are:

DGH	Meaning
0b0000	Data Gathering Hint is not implemented.
0b0001	Data Gathering Hint is implemented.

All other values are reserved.

FEAT_DGH implements the functionality identified by 0b0001.

From Armv8.0, the permitted values are 0b0000 and 0b0001.

If the DGH instruction has no effect in preventing the merging of memory accesses, the value of this field is 0b0000.

BF16, bits [47:44]

Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are:

BF16	Meaning
0b0000	BFloat16 instructions are not implemented.
0b0001	BFCVT, BFCVTN, BFCVTN2, BFDOT, BFMLALB, BFMLALT, and BFMMMLA instructions are implemented.
0b0010	As 0b0001, but the FPCR .EBF field is also supported.

All other values are reserved.

Otherwise, from Armv8.6, the only permitted value is 0b0001.

FEAT_BF16 **adds** the functionality identified by 0b0001.

FEAT_EBF16 **adds** the functionality identified by 0b0010.

When ~~FEAT_SVE or FEAT_SME are implemented, this field must return the same value as~~ **FEAT_SVE or FEAT_SME is implemented, this field must return the same value as** ID_AA64ZFR0_EL1.BF16.

From Armv8.6 and Armv9.1, the value 0b0000 is not permitted.

~~If FEAT_SME is implemented, the permitted values are 0b0001 and 0b0010.~~

SPECRES, bits [43:40]

Indicates support for prediction invalidation instructions in AArch64 state. Defined values are:

SPECRES	Meaning
0b0000	Prediction CFP invalidation RCTX, DVP RCTX, and CPP RCTX instructions are not implemented.
0b0001	CFP RCTX, DVP RCTX and CPP RCTX CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.
0b0010	As 0b0001, and COSP RCTX instruction is implemented.

All other values are reserved.

FEAT_SPECRES implements the functionality identified by 0b0001.

FEAT_SPECRES2 implements the functionality identified by 0b0010.

~~In Armv8.0, the permitted values are 0b0000 and 0b0001.~~

From Armv8.5, the ~~only permitted value is~~ **0b0000** ~~0b0001 is not permitted.~~

From Armv8.9, the value 0b0001 is not permitted.

SB, bits [39:36]

Indicates support for SB instruction in AArch64 state. Defined values are:

SB	Meaning
0b0000	SB instruction is not implemented.
0b0001	SB instruction is implemented.

All other values are reserved.

FEAT_SB implements the functionality identified by 0b0001.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

FRINTTS, bits [35:32]

Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are:

FRINTTS	Meaning
0b0000	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are not implemented.
0b0001	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.

All other values are reserved.

FEAT_FRINTTS implements the functionality identified by 0b0001.

From Armv8.5, the only permitted value is 0b0001.

GPI, bits [31:28]

Indicates support for an IMPLEMENTATION DEFINED algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:

GPI	Meaning
0b0000	Generic Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.
0b0001	Generic Authentication using an IMPLEMENTATION DEFINED algorithm is implemented. This includes the PACGA instruction.

All other values are reserved.

FEAT_PACIMP implements the functionality identified by 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

If the value of ID_AA64ISAR1_EL1.GPA is non-zero, or the value of [ID_AA64ISAR2_EL1.GPA3](#) is non-zero, this field must have the value 0b0000.

GPA, bits [27:24]

Indicates whether the QARMA5 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:

GPA	Meaning
0b0000	Generic Authentication using the QARMA5 algorithm is not implemented.
0b0001	Generic Authentication using the QARMA5 algorithm is implemented. This includes the PACGA instruction.

All other values are reserved.

FEAT_PACQARMA5 implements the functionality identified by 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

If the value of ID_AA64ISAR1_EL1.GPI is non-zero, or the value of [ID_AA64ISAR2_EL1.GPA3](#) is non-zero, this field must have the value 0b0000.

LRCPC, bits [23:20]

Indicates support for weaker release consistency, RCpc, based model. Defined values are:

LRCPC	Meaning
0b0000	RCpc The LDAPR*, LDAPUR*, and STLUR* instructions are not implemented.
0b0001	The LDAPUR*, and STLUR* instructions are not implemented. The no offset LDAPR, LDAPRB, and LDAPRH* instructions are implemented.
0b0010	As The LDAPR*, LDAPUR*, and STLUR* instructions are implemented. 0b0001, and the LDAPR (unscaled immediate) and STLR (unscaled immediate) instructions are implemented.
0b0011	As 0b0010, and the post-index LDAPR, LDIAPP, STILP, and pre-index STLR instructions are implemented. If Advanced SIMD and floating-point is implemented, then the LDAPUR (SIMD&FP), LDAP1 (SIMD&FP), STLUR (SIMD&FP), and STL1 (SIMD&FP) instructions are implemented in Advanced SIMD and floating-point.

All other values are reserved.

FEAT_LRCPC implements the functionality identified by the value 0b0001.

FEAT_LRCPC2 implements the functionality identified by the value 0b0010.

FEAT_LRCPC3 implements the functionality identified by the value 0b0011.

In Armv8.2, the permitted values are 0b0000, 0b0001, and 0b0010.

From Armv8.3, the value 0b0000 is not permitted.

In Armv8.3, the permitted values are 0b0001 and 0b0010.

From Armv8.4, the only permitted value is 0b0001. 0b0010 is not permitted.

FCMA, bits [19:16]

Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are:

FCMA	Meaning
0b0000	The FCMLA and FCADD instructions are not implemented.
0b0001	The FCMLA and FCADD instructions are implemented.

All other values are reserved.

FEAT_FCMA implements the functionality identified by the value 0b0001.

In Armv8.0, Armv8.1, and Armv8.2, the only permitted value is 0b0000.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is 0b0001.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is 0b0000.

JSCVT, bits [15:12]

Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are:

JSCVT	Meaning
0b0000	The FJCVTZS instruction is not implemented.
0b0001	The FJCVTZS instruction is implemented.

All other values are reserved.

FEAT_JSCVT implements the functionality identified by 0b0001.

In Armv8.0, Armv8.1, and Armv8.2, the only permitted value is 0b0000.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is 0b0001.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is 0b0000.

API, bits [11:8]

Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are:

API	Meaning
0b0000	Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.
0b0001	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented, with the HaveEnhancedPAC() and HaveEnhancedPAC2() functions returning FALSE.
0b0010	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented, with the HaveEnhancedPAC() function returning TRUE, and the HaveEnhancedPAC2() function returning FALSE.
0b0011	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.
0b0100	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
0b0101	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

All other values are reserved.

FEAT_PAuth implements the functionality identified by 0b0001.

FEAT_EPAC implements the functionality identified by 0b0010.

FEAT_PAuth2 implements the functionality identified by 0b0011.

FEAT_FPAC implements the functionality identified by 0b0100.

FEAT_FPACCOMBINE implements the functionality identified by 0b0101.

When this field is non-zero, FEAT_PACIMP is implemented.

In Armv8.3, the permitted values are 0b0001, 0b0010, 0b0011, 0b0100, and 0b0101.

From Armv8.6, the permitted values are 0b0011, 0b0100, and 0b0101.

If the value of ID_AA64ISAR1_EL1.APA is non-zero, or the value of [ID_AA64ISAR2_EL1.APA3](#) is non-zero, this field must have the value 0b0000.

APA, bits [7:4]

Indicates whether the QARMA5 algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are:

APA	Meaning
0b0000	Address Authentication using the QARMA5 algorithm is not implemented.
0b0001	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC() and HaveEnhancedPAC2() functions returning FALSE.
0b0010	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC() function returning TRUE and the HaveEnhancedPAC2() function returning FALSE.
0b0011	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning FALSE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
0b0100	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
0b0101	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

All other values are reserved.

FEAT_PAuth implements the functionality identified by 0b0001.

FEAT_EPAC implements the functionality identified by 0b0010.

FEAT_PAuth2 implements the functionality identified by 0b0011.

FEAT_FPAC implements the functionality identified by 0b0100.

FEAT_FPACCOMBINE implements the functionality identified by 0b0101.

When this field is non-zero, FEAT_PACQARMA5 is implemented.

In Armv8.3, the permitted values are 0b0001, 0b0010, 0b0011, 0b0100, and 0b0101.

From Armv8.6, the permitted values are 0b0011, 0b0100, and 0b0101.

If the value of ID_AA64ISAR1_EL1.API is non-zero, or the value of [ID_AA64ISAR2_EL1.APA3](#) is non-zero, this field must have the value 0b0000.

DPB, bits [3:0]

Data Persistence writeback. Indicates support for the [DC CVAP](#) and [DC CVADP](#) instructions in AArch64 state. Defined values are:

DPB	Meaning
0b0000	DC CVAP not supported.
0b0001	DC CVAP supported.
0b0010	DC CVAP and DC CVADP supported.

All other values are reserved.

FEAT_DPB implements the functionality identified by the value 0b0001.

FEAT_DPB2 implements the functionality identified by the value 0b0010.

In Armv8.2, the permitted values are 0b0001 and 0b0010.

From Armv8.5, the only permitted value is 0b0010.

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

(old)

htmldiff from-

(new)

ID_AA64ISAR2_EL1, AArch64 Instruction Set Attribute Register 2

The ID_AA64ISAR2_EL1 characteristics are:

Purpose

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_AA64ISAR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								CSSC				RPRFM				RES0				PRFM SLC				SYSINSTR_128				SYSREG_128			
CLRBHB								PAC frac				BC				MOPS				APA3				GPA3				RPREs			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:56:28]

Reserved, RES0.

CSSC, bits [55:52]

Indicates support for common short sequence compression instructions. Defined values are:

CSSC	Meaning
0b0000	Common short sequence compression instructions are not implemented.
0b0001	Common short sequence compression instructions are implemented.

All other values are reserved.

FEAT_CSSC adds the functionality identified by the value 0b0001.

From Armv9.4, the value 0b0000 is not permitted.

RPRFM, bits [51:48]

RPRFM hint instruction. Defined values are:

RPRFM	Meaning
0b0000	RPRFM hint instruction is not implemented and is treated as a NOP.
0b0001	RPRFM hint instruction is implemented.

All other values are reserved.

FEAT_RPRFM adds the functionality identified by the value 0b0001.

Bits [47:44]

Reserved, RES0.

PRFMSLC, bits [43:40]

Indicates whether the PFRM instructions support a system level cache option. Defined values are:

PRFMSLC	Meaning
0b0000	The PRFM instructions do not support the SLC target.
0b0001	The PRFM instructions support the SLC target.

All other values are reserved.

FEAT_PRFMSLC implements the functionality identified by 0b0001.

SYSINSTR_128, bits [39:36]

SYSINSTR_128. Defined values are:

SYSINSTR_128	Meaning
0b0000	System instructions that take 128-bit inputs are not supported.
0b0001	System instructions that take 128-bit inputs are supported.

All other values are reserved.

FEAT_SYSINSTR128 implements the functionality identified by 0b0001.

SYSREG_128, bits [35:32]

SYSREG_128. Defined values are:

SYSREG_128	Meaning
0b0000	Instructions to access 128-bit System Registers are not supported.
0b0001	Instructions to access 128-bit System Registers are supported.

All other values are reserved.

FEAT_SYSREG128 implements the functionality identified by 0b0001.

CLRBHB, bits [31:28]

Indicates support for the CLRBHB instruction in AArch64 state. Defined values are:

CLRBHB	Meaning
0b0000	CLRBHB instruction is not implemented.
0b0001	CLRBHB instruction is implemented.

All other values are reserved.

FEAT_CLRBHB implements the functionality identified by 0b0001.

From Armv8.9, the value 0b0000 is not permitted.

PAC_frac, bits [27:24]

Indicates whether the ConstPACField() function used as part of the PAC addition returns FALSE or TRUE.

PAC_frac	Meaning
0b0000	ConstPACField() returns FALSE.
0b0001	ConstPACField() returns TRUE.

All other values are reserved.

FEAT_CONSTPACFIELD implements the functionality identified by 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

BC, bits [23:20]

Indicates support for the BC instruction in AArch64 state. Defined values are:

BC	Meaning
0b0000	BC instruction is not implemented.
0b0001	BC instruction is implemented.

All other values are reserved.

FEAT_HBC implements the functionality identified by the value 0b0001.

From Armv8.8, the only permitted value is 0b0001.

MOPS, bits [19:16]

Indicates support for the Memory Copy and Memory Set instructions in AArch64 state.

MOPS	Meaning
0b0000	The Memory Copy and Memory Set instructions are not implemented in AArch64 state.
0b0001	The Memory Copy and Memory Set instructions are implemented in AArch64 state with the following exception. If FEAT_MTE is implemented, then SETGP*, SETGM* and SETGE* instructions are also supported.

All other values are reserved.

FEAT_MOPS implements the functionality identified by the value 0b0001.

From Armv8.8, the only permitted value is 0b0001.

APA3, bits [15:12]

Indicates whether the QARMA3 algorithm is implemented in the PE for address authentication in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are:

APA3	Meaning
0b0000	Address Authentication using the QARMA3 algorithm is not implemented.
0b0001	Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC() and HaveEnhancedPAC2() functions returning FALSE.
0b0010	Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC() function returning TRUE and the HaveEnhancedPAC2() function returning FALSE.
0b0011	Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning FALSE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
0b0100	Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
0b0101	Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

All other values are reserved.

FEAT_PAuth implements the functionality identified by 0b0001.

FEAT_EPAC implements the functionality identified by 0b0010.

FEAT_PAuth2 implements the functionality identified by 0b0011.

FEAT_FPAC implements the functionality identified by 0b0100.

FEAT_FPACCOMBINE implements the functionality identified by 0b0101.

When this field is non-zero, FEAT_PACQARMA3 is implemented.

In Armv8.3, the permitted values are 0b0000, 0b0001, 0b0010, 0b0011, 0b0100, and 0b0101.

From Armv8.6, the permitted values are 0b0011, 0b0100, and 0b0101.

If the value of [ID_AA64ISAR1_EL1.API](#) is non-zero, or the value of [ID_AA64ISAR1_EL1.APA](#) is non-zero, this field must have the value 0b0000.

GPA3, bits [11:8]

Indicates whether the QARMA3 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:

GPA3	Meaning
0b0000	Generic Authentication using the QARMA3 algorithm is not implemented.
0b0001	Generic Authentication using the QARMA3 algorithm is implemented. This includes the PACGA instruction.

All other values are reserved.

FEAT_PACQARMA3 implements the functionality identified by 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

If the value of [ID_AA64ISAR1_EL1.GPI](#) is non-zero, or the value of [ID_AA64ISAR1_EL1.GPA](#) is non-zero, this field must have the value 0b0000.

RPRES, bits [7:4]

Indicates support for 12 bits of mantissa in reciprocal and reciprocal square root instructions in AArch64 state, when [FPCR.AH](#) is 1. Defined values are:

RPRES	Meaning	Applies when
0b0000	Reciprocal and reciprocal square root estimates give 8 bits of mantissa, when FPCR.AH is 1.	When FPCR.AH == 1
0b0001	Reciprocal and reciprocal square root estimates give 12 bits of mantissa, when FPCR.AH is 1.	When FPCR.AH == 1

All other values are reserved.

FEAT_RPRES implements the functionality identified by the value 0b0001.

From Armv8.7, if Advanced SIMD and floating-point is implemented, the only permitted value is 0b0001.

WFxT, bits [3:0]

Indicates support for the WFET and WFIT instructions in AArch64 state. Defined values are:

WFxT	Meaning
0b0000	WFET and WFIT are not supported.
0b0010	WFET and WFIT are supported, and the register number is reported in the ESR ELx on exceptions.

All other values are reserved.

FEAT_WFxT implements the functionality identified by the value 0b0010.

From Armv8.7, the only permitted value is 0b0010.

Accessing ID_AA64ISAR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_AA64ISAR2_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_AA64ISAR2_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR2_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR2_EL1;

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

The ID_AA64MMFR0_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64MMFR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ECV				FGT				RES0								ExS				TGran4_2				TGran64_2				TGran16_2			
TGran4				TGran64				TGran16				BigEndEL0				SNSMem				BigEnd				ASIDBits				PARange			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ECV, bits [63:60]

Indicates presence of Enhanced Counter Virtualization. Defined values are:

ECV	Meaning
0b0000	Enhanced Counter Virtualization is not implemented.
0b0001	Enhanced Counter Virtualization is implemented. Supports CNTHCTL_EL2 .{EL1TVT, EL1TVCT, EL1NVPCT, EL1NVVCT, EVNTIS}, CNTKCTL_EL1 .EVNTIS, CNTPTSS_EL0 counter views, and CNTVCTSS_EL0 counter views. Extends the PMSCR_EL1 .PCT, PMSCR_EL2 .PCT, TRFCR_EL1 .TS, and TRFCR_EL2 .TS fields.
0b0010	As 0b0001, and also includes support for CNTHCTL_EL2 .ECV and CNTPOFF_EL2 .

All other values are reserved.

FEAT_ECV implements the functionality identified by the values 0b0001 and 0b0010.

From Armv8.6, the only permitted values are 0b0001 and 0b0010.

FGT, bits [59:56]

Indicates presence of the Fine-Grained Trap controls. Defined values are:

FGT	Meaning
0b0000	Fine-grained trap controls are not implemented.
0b0001	Fine-grained trap controls are implemented. Supports: <ul style="list-style-type: none"> If EL2 is implemented, the HAFGRTR_EL2, HDFGRTR_EL2, HDFGWTR_EL2, HFGTRTR_EL2, HFGITR_EL2 and HFGWTR_EL2 registers, and their associated traps. If EL2 is implemented, MDCR_EL2.TDCC. If EL3 is implemented, MDCR_EL3.TDCC. If both EL2 and EL3 are implemented, SCR_EL3.FGTEn.
0b0010	As 0b0001, and also includes support for: <ul style="list-style-type: none"> If EL2 is implemented, the HDFGRTR2_EL2, HDFGWTR2_EL2, HFGITR2_EL2, HFGTRTR2_EL2, and HFGWTR2_EL2 registers, and their associated traps. If both EL2 and EL3 are implemented, SCR_EL3.FGTEn2.

All other values are reserved.

FEAT_FGT implements the functionality identified by the value 0b0001.

FEAT_FGT2 implements the functionality identified by the value 0b0010.

From Armv8.6, the value 0b0000 is not permitted.

From Armv8.9, the value 0b0001 is not permitted.

Bits [55:48]

Reserved, RES0.

ExS, bits [47:44]

Indicates support for disabling context synchronizing exception entry and exit. Defined values are:

ExS	Meaning
0b0000	All exception entries and exits are context synchronization events.
0b0001	Non-context synchronizing exception entry and exit are supported.

All other values are reserved.

FEAT_ExS implements the functionality identified by the value 0b0001.

TGran4_2, bits [43:40]

Indicates support for 4KB memory granule size at stage 2. Defined values are:

TGran4_2	Meaning	Applies when
0b0000	Support for 4KB granule at stage 2 is identified in the ID_AA64MMFR0_EL1.TGran4 field.	
0b0001	4KB granule not supported at stage 2.	
0b0010	4KB granule supported at stage 2.	
0b0011	4KB granule at stage 2 supports 52-bit input addresses and can describe 52-bit output addresses.	When FEAT_LPA2 is implemented

All other values are reserved.

The 0b0000 value is deprecated.

Note

This field does not follow the standard ID scheme. See Alternative ID scheme used for ID_AA64MMFR0_EL1 stage 2 granule sizes for more information.

TGran64_2, bits [39:36]

Indicates support for 64KB memory granule size at stage 2. Defined values are:

TGran64_2	Meaning
0b0000	Support for 64KB granule at stage 2 is identified in the ID_AA64MMFR0_EL1.TGran64 field.
0b0001	64KB granule not supported at stage 2.
0b0010	64KB granule supported at stage 2.

All other values are reserved.

The 0b0000 value is deprecated.

Note

This field does not follow the standard ID scheme. See Alternative ID scheme used for ID_AA64MMFR0_EL1 stage 2 granule sizes for more information.

TGran16_2, bits [35:32]

Indicates support for 16KB memory granule size at stage 2. Defined values are:

TGran16_2	Meaning	Applies when
0b0000	Support for 16KB granule at stage 2 is identified in the ID_AA64MMFR0_EL1.TGran16 field.	
0b0001	16KB granule not supported at stage 2.	
0b0010	16KB granule supported at stage 2.	
0b0011	16KB granule at stage 2 supports 52-bit input addresses and can describe 52-bit output addresses.	When FEAT_LPA2 is implemented

All other values are reserved.

The 0b0000 value is deprecated.

Note

This field does not follow the standard ID scheme. See Alternative ID scheme used for ID_AA64MMFR0_EL1 stage 2 granule sizes for more information.

TGran4, bits [31:28]

Indicates support for 4KB memory translation granule size. Defined values are:

TGran4	Meaning	Applies when
0b0000	4KB granule supported.	
0b0001	4KB granule supports 52-bit input addresses and can describe 52-bit output addresses.	When FEAT_LPA2 is implemented
0b1111	4KB granule not supported.	

All other values are reserved.

TGran64, bits [27:24]

Indicates support for 64KB memory translation granule size. Defined values are:

TGran64	Meaning
0b0000	64KB granule supported.
0b1111	64KB granule not supported.

All other values are reserved.

TGran16, bits [23:20]

Indicates support for 16KB memory translation granule size. Defined values are:

TGran16	Meaning	Applies when
0b0000	16KB granule not supported.	
0b0001	16KB granule supported.	
0b0010	16KB granule supports 52-bit input addresses and can describe 52-bit output addresses.	When FEAT_LPA2 is implemented

All other values are reserved.

BigEndEL0, bits [19:16]

Indicates support for mixed-endian at EL0 only. Defined values are:

BigEndEL0	Meaning
0b0000	No mixed-endian support at EL0. The SCTLR_EL1.E0E bit has a fixed value.
0b0001	Mixed-endian support at EL0. The SCTLR_EL1.E0E bit can be configured.

All other values are reserved.

This field is invalid and is RES0 if ID_AA64MMFR0_EL1.BigEnd is not 0b0000.

SNSMem, bits [15:12]

Indicates support for a distinction between Secure and Non-secure Memory. Defined values are:

SNSMem	Meaning
0b0000	Does not support a distinction between Secure and Non-secure Memory.
0b0001	Does support a distinction between Secure and Non-secure Memory.

Note

If EL3 is implemented, the value 0b0000 is not permitted.

All other values are reserved.

BigEnd, bits [11:8]

Indicates support for mixed-endian configuration. Defined values are:

BigEnd	Meaning
0b0000	No mixed-endian support. The SCTL _R _EL _x .EE bits have a fixed value. See the BigEndEL0 field, bits[19:16], for whether EL0 supports mixed-endian.
0b0001	Mixed-endian support. The SCTL _R _EL _x .EE and SCTL_R_EL1.E0E bits can be configured.

All other values are reserved.

ASIDBits, bits [7:4]

Number of ASID bits. Defined values are:

ASIDBits	Meaning
0b0000	8 bits.
0b0010	16 bits.

All other values are reserved.

PARange, bits [3:0]

Physical Address range supported. Defined values are:

PARange	Meaning	Applies when
0b0000	32 bits, 4GB.	
0b0001	36 bits, 64GB.	
0b0010	40 bits, 1TB.	
0b0011	42 bits, 4TB.	
0b0100	44 bits, 16TB.	
0b0101	48 bits, 256TB.	
0b0110	52 bits, 4PB.	When FEAT_LPA is implemented or FEAT_LPA2 is implemented
0b0111	56 bits, 64PB.	When FEAT_D128 is implemented

All other values are reserved.

The value 0b0110 is permitted only if the implementation includes FEAT_LPA, otherwise it is reserved.

Accessing ID_AA64MMFR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64MMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_AA64MMFR0_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = ID_AA64MMFR0_EL1;
    elseif PSTATE.EL == EL3 then
        X[t, 64] = ID_AA64MMFR0_EL1;

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

The ID_AA64MMFR1_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64MMFR1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ECBHBRES0				CMOW				TIDCP1				nTLBPA				AFP				HCX				ETS				TWED			
XNX				SpecSEI				PAN				LO				HPDS				VH				VMIDBits				HAFDBS			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ECBHB, bitsBits [63:60]

Indicates support for cache maintenance instruction permission. Defined values are:

Reserved, RES0.

ECBHB	Meaning
0b0000	The implementation does not disclose whether the branch history information created in a context before an exception to a higher exception level using AArch64 can be used by code before that exception to exploitatively control the execution of any indirect branches in code in a different context after the exception.
0b0001	The branch history information created in a context before an exception to a higher exception level using AArch64 cannot be used by code before that exception to exploitatively control the execution of any indirect branches in code in a different context after the exception.

All other values are reserved.

FEAT_ECBHB implements the functionality identified by the value 0b0001.

From Armv8.9, the value 0b0000 is not permitted.

CMOW, bits [59:56]

Indicates support for cache maintenance instruction permission. Defined values are:

CMOW	Meaning
0b0000	SCTLR_EL1 .CMOW, SCTLR_EL2 .CMOW, and HCRX_EL2 .CMOW bits are not implemented.
0b0001	SCTLR_EL1 .CMOW is implemented. If EL2 is implemented, SCTLR_EL2 .CMOW and HCRX_EL2 .CMOW bits are implemented.

All other values are reserved.

FEAT_CMOW implements the functionality identified by the value 0b0001.

From Armv8.8, the only permitted value is 0b0001.

TIDCP1, bits [55:52]

Indicates whether [SCTLR_EL1](#).TIDCP and [SCTLR_EL2](#).TIDCP are implemented in AArch64 state. Defined values are:

TIDCP1	Meaning
0b0000	SCTLR_EL1 .TIDCP and SCTLR_EL2 .TIDCP bits are not implemented and are RES0.
0b0001	SCTLR_EL1 .TIDCP bit is implemented. If EL2 is implemented, SCTLR_EL2 .TIDCP bit is implemented.

All other values are reserved.

FEAT_TIDCP1 implements the functionality identified by the value 0b0001.

From Armv8.8, the only permitted value is 0b0001.

nTLBPA, bits [51:48]

Indicates support for intermediate caching of translation table walks. Defined values are:

nTLBPA	Meaning
0b0000	The intermediate caching of translation table walks might include non-coherent physical translation caches.
0b0001	The intermediate caching of translation table walks does not include non-coherent physical translation caches.

Non-coherent physical translation caches are non-coherent caches of previous valid translation table entries since the last completed relevant TLBI applicable to the PE, where either:

- The caching is indexed by the physical address of the location holding the translation table entry.
- The caching is used for stage 1 translations and is indexed by the intermediate physical address of the location holding the translation table entry.

All other values are reserved.

FEAT_nTLBPA implements the functionality identified by the value 0b0001.

From Armv8.0, the permitted values are 0b0000 and 0b0001.

AFP, bits [47:44]

Indicates support for [FPCR](#).{AH, FIZ, NEP}. Defined values are:

AFP	Meaning
0b0000	The FPCR .{AH, FIZ, NEP} fields are not supported.
0b0001	The FPCR .{AH, FIZ, NEP} fields are supported.

All other values are reserved.

FEAT_AFP implements the functionality identified by the value 0b0001.

From Armv8.7, if Advanced SIMD and floating-point is implemented, the only permitted value is 0b0001.

HCX, bits [43:40]

Indicates support for [HCRX_EL2](#) and its associated EL3 trap. Defined values are:

HCX	Meaning
0b0000	HCRX_EL2 and its associated EL3 trap are not supported.
0b0001	HCRX_EL2 and its associated EL3 trap are supported.

All other values are reserved.

FEAT_HCX implements the functionality identified by the value 0b0001.

From Armv8.7, if EL2 is implemented, the only permitted value is 0b0001.

ETS, bits [39:36]

Indicates support for Enhanced Translation Synchronization. Defined values are:

ETS	Meaning
0b0000	Enhanced Translation Synchronization is not supported.
0b0001	Enhanced Translation Synchronization is supported.

All other values are reserved.

FEAT_ETS implements the functionality identified by the value 0b0001.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.7, the only permitted value is 0b0001.

TWED, bits [35:32]

Indicates support for the configurable delayed trapping of WFE. Defined values are:

TWED	Meaning
0b0000	Configurable delayed trapping of WFE is not supported.
0b0001	Configurable delayed trapping of WFE is supported.

All other values are reserved.

FEAT_TWED implements the functionality identified by the value 0b0001.

From Armv8.6, the permitted values are 0b0000 and 0b0001.

XNX, bits [31:28]

Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are:

XNX	Meaning
0b0000	Distinction between EL0 and EL1 execute-never control at stage 2 not supported.
0b0001	Distinction between EL0 and EL1 execute-never control at stage 2 supported.

All other values are reserved.

FEAT_XNX implements the functionality identified by the value 0b0001.

From Armv8.2, the only permitted value is 0b0001.

SpecSEI, bits [27:24]**When FEAT_RAS is implemented:**

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches.

SpecSEI	Meaning
0b0000	The PE never generates an SError interrupt due to an External abort on a speculative read.
0b0001	The PE might generate an SError interrupt due to an External abort on a speculative read.

All other values are reserved.

Otherwise:

Reserved, RES0.

PAN, bits [23:20]

Privileged Access Never. Indicates support for the PAN bit in PSTATE, [SPSR_EL1](#), [SPSR_EL2](#), [SPSR_EL3](#), and [DSPSR_EL0](#). Defined values are:

PAN	Meaning
0b0000	PAN not supported.
0b0001	PAN supported.
0b0010	PAN supported and AT S1E1RP and AT S1E1WP instructions supported.
0b0011	PAN supported, AT S1E1RP and AT S1E1WP instructions supported, and SCTLR_EL1 .EPAN and SCTLR_EL2 .EPAN bits supported.

All other values are reserved.

FEAT_PAN implements the functionality identified by the value 0b0001.

FEAT_PAN2 implements the functionality added by the value 0b0010.

FEAT_PAN3 implements the functionality added by the value 0b0011.

In Armv8.1, the permitted values are 0b0001, 0b0010, and 0b0011.

From Armv8.2, the permitted values are 0b0010 and 0b0011.

From Armv8.7, the only permitted value is 0b0011.

LO, bits [19:16]

LORegions. Indicates support for LORegions. Defined values are:

LO	Meaning
0b0000	LORegions not supported.
0b0001	LORegions supported.

All other values are reserved.

FEAT_LOR implements the functionality identified by the value 0b0001.

From Armv8.1, the only permitted value is 0b0001.

HPDS, bits [15:12]

Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are:

HPDS	Meaning
0b0000	Disabling of hierarchical controls not supported.
0b0001	Disabling of hierarchical controls supported with the TCR_EL1 .{HPD1, HPD0}, TCR_EL2 .HPD or TCR_EL2 .{HPD1, HPD0}, and TCR_EL3 .HPD bits.
0b0010	As for value 0b0001, and adds possible hardware allocation of bits[62:59] of the Translation table descriptors from the final lookup level for IMPLEMENTATION DEFINED use.

All other values are reserved.

FEAT_HPDS implements the functionality identified by the value 0b0001.

FEAT_HPDS2 implements the functionality identified by the value 0b0010.

From Armv8.1, the value 0b0000 is not permitted.

VH, bits [11:8]

Virtualization Host Extensions. Defined values are:

VH	Meaning
0b0000	Virtualization Host Extensions not supported.
0b0001	Virtualization Host Extensions supported.

All other values are reserved.

FEAT_VHE implements the functionality identified by the value 0b0001.

From Armv8.1, the only permitted value is 0b0001.

VMIDBits, bits [7:4]

Number of VMID bits. Defined values are:

VMIDBits	Meaning
0b0000	8 bits
0b0010	16 bits

All other values are reserved.

FEAT_VMID16 implements the functionality identified by the value 0b0010.

From Armv8.1, the permitted values are 0b0000 and 0b0010.

HAFDBS, bits [3:0]

Hardware updates to Access flag and Dirty state in translation tables. Defined values are:

HAFDBS	Meaning
0b0000	Hardware update of the Access flag and dirty state are not supported.
0b0001	Hardware update of the Access flag is supported.
0b0010	Hardware update of both the Access flag and dirty state is supported.

All other values are reserved.

FEAT_HAFDBS implements the functionality identified by the values 0b0001 and 0b0010.

From Armv8.1, the permitted values are 0b0000, 0b0001, and 0b0010.

Accessing ID_AA64MMFR1_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64MMFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR1_EL1;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

The ID_AA64MMFR2_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_AA64MMFR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
EOPD				EVT				BBM				TTL				RES0				FWB				IDS				AT			
ST				NV				CCIDX				VARange				IESB				LSM				UAO				CnP			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EOPD, bits [63:60]

Indicates support for the EOPD mechanism. Defined values are:

EOPD	Meaning
0b0000	EOPDx mechanism is not implemented.
0b0001	EOPDx mechanism is implemented.

All other values are reserved.

FEAT_EOPD implements the functionality identified by the value 0b0001.

In Armv8.4, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

If FEAT_EOPD is implemented, FEAT_CSV3 must be implemented.

EVT, bits [59:56]

Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the [HCR_EL2](#).{TTLBOS, TLBIS, TOCU, TICAB, TID4} traps. Defined values are:

EVT	Meaning
0b0000	HCR_EL2 .{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are not supported.
0b0001	HCR_EL2 .{TOCU, TICAB, TID4} traps are supported. HCR_EL2 .{TTLBOS, TTLBIS} traps are not supported.
0b0010	HCR_EL2 .{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.

All other values are reserved.

FEAT_EVT implements the functionality identified by the values 0b0001 and 0b0010.

If EL2 is not implemented, the only permitted value is 0b0000.

In Armv8.2, the permitted values are 0b0000, 0b0001, and 0b0010.

From Armv8.5, the permitted values are:

- 0b0000 when EL2 is not implemented.
- 0b0010 when EL2 is implemented.

BBM, bits [55:52]

Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation.

BBM	Meaning
0b0000	Level 0 support for changing block size is supported.
0b0001	Level 1 support for changing block size is supported.
0b0010	Level 2 support for changing block size is supported.

All other values are reserved.

FEAT_BBM implements the functionality identified by the values 0b0000, 0b0001, and 0b0010.

From Armv8.4, the permitted values are 0b0000, 0b0001, and 0b0010.

TTL, bits [51:48]

Indicates support for TTL field in address operations. Defined values are:

TTL	Meaning
0b0000	TLB maintenance instructions by address have bits[47:44] as RES0.
0b0001	TLB maintenance instructions by address have bits[47:44] holding the TTL field.

All other values are reserved.

FEAT_TTL implements the functionality identified by the value 0b0001.

This field affects [TLBI IPAS2E1](#), [TLBI IPAS2E1IS](#), [TLBI IPAS2E1OS](#), [TLBI IPAS2LE1](#), [TLBI IPAS2LE1IS](#), [TLBI IPAS2LE1OS](#), [TLBI VAAE1](#), [TLBI VAAE1IS](#), [TLBI VAAE1OS](#), [TLBI VAALE1](#), [TLBI VAALE1IS](#), [TLBI VAALE1OS](#), [TLBI VAE1](#), [TLBI VAE1IS](#), [TLBI VAE1OS](#), [TLBI VAE2](#), [TLBI VAE2IS](#), [TLBI VAE2OS](#), [TLBI VAE3](#), [TLBI VAE3IS](#), [TLBI VAE3OS](#), [TLBI VALE1](#), [TLBI VALE1IS](#), [TLBI VALE1OS](#), [TLBI VALE2](#), [TLBI VALE2IS](#), [TLBI VALE2OS](#), [TLBI VALE3](#), [TLBI VALE3IS](#), [TLBI VALE3OS](#).

From Armv8.4, the only permitted value is 0b0001.

Bits [47:44]

Reserved, RES0.

FWB, bits [43:40]

Indicates support for [HCR_EL2](#).FWB. Defined values are:

FWB	Meaning
0b0000	HCR_EL2 .FWB bit is not supported.
0b0001	HCR_EL2 .FWB is supported.

All other values reserved.

FEAT_S2FWB implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

IDS, bits [39:36]

Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are:

IDS	Meaning
0b0000	An exception which is generated by a read access to the feature ID space, other than a trap caused by HCR_EL2 .TIDx, SCTLR_EL1 .UCT, or SCTLR_EL2 .UCT, is reported by ESR_ELx.EC == 0x0.
0b0001	All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.

All other values are reserved.

The Feature ID space is defined as the System register space in AArch64 with op0==3, op1=={0, 1, 3}, CRn==0, CRm=={0-7}, op2=={0-7}.

FEAT_IDST implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

AT, bits [35:32]

Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are:

AT	Meaning
0b0000	Unaligned single-copy atomicity and atomic functions are not supported.
0b0001	Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.

All other values are reserved.

FEAT_LSE2 implements the functionality identified by the value 0b0001.

In Armv8.2, the permitted values are 0b0000 and 0b0001.

From Armv8.4, the only permitted value is 0b0001.

ST, bits [31:28]

Identifies support for small translation tables. Defined values are:

ST	Meaning
0b0000	The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 39.
0b0001	The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.

All other values are reserved.

FEAT_TTST implements the functionality identified by the value 0b0001.

If FEAT_SEL2 is implemented, the only permitted value is 0b0001.

In an implementation which does not support FEAT_SEL2, the permitted values are 0b0000 and 0b0001.

NV, bits [27:24]

Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are:

NV	Meaning
0b0000	Nested virtualization is not supported.
0b0001	The HCR_EL2 .{AT, NV1, NV} bits are implemented.
0b0010	The VNCR_EL2 register and the HCR_EL2 .{NV2, AT, NV1, NV} bits are implemented.

All other values are reserved.

If EL2 is not implemented, the only permitted value is 0b0000.

FEAT_NV implements the functionality identified by the value 0b0001.

FEAT_NV2 implements the functionality identified by the value 0b0010.

In Armv8.3, if EL2 is implemented, the permitted values are 0b0000 and 0b0001.

From Armv8.4, if EL2 is implemented, the permitted values are 0b0000, 0b0001, and 0b0010.

CCIDX, bits [23:20]

Support for the use of revised [CCSIDR_EL1](#) register format. Defined values are:

CCIDX	Meaning
0b0000	32-bit format implemented for all levels of the CCSIDR_EL1.
0b0001	64-bit format implemented for all levels of the CCSIDR_EL1.

All other values are reserved.

FEAT_CCIDX implements the functionality identified by the value 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

VARange, bits [19:16]

Indicates support for a larger virtual address. Defined values are:

VARange	Meaning	Applies when
0b0000	VMSAv8-64 supports 48-bit VAs.	
0b0001	VMSAv8-64 supports 52-bit VAs when using the 64KB translation granule. The size for other translation granules is not defined by this field.	
0b0010	VMSAv9-128 supports 56-bit VAs.	When FEAT_D128 is implemented

All other values are reserved.

FEAT_LVA implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

IESB, bits [15:12]

Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are:

IESB	Meaning
0b0000	IESB bit in the SCTLR_ELx registers is not supported.
0b0001	IESB bit in the SCTLR_ELx registers is supported.

All other values are reserved.

FEAT_IESB implements the functionality identified by the value 0b0001.

LSM, bits [11:8]

Indicates support for LSMAOE and nTLSMD bits in [SCTLR_EL1](#) and [SCTLR_EL2](#). Defined values are:

LSM	Meaning
0b0000	LSMAOE and nTLSMD bits not supported.
0b0001	LSMAOE and nTLSMD bits supported.

All other values are reserved.

FEAT_LSMAOC implements the functionality identified by the value 0b0001.

UAO, bits [7:4]

User Access Override. Defined values are:

UAO	Meaning
0b0000	UAO not supported.
0b0001	UAO supported.

All other values are reserved.

FEAT_UAO implements the functionality identified by the value 0b0001.

From Armv8.2, the only permitted value is 0b0001.

CnP, bits [3:0]

Indicates support for Common not Private translations. Defined values are:

CnP	Meaning
0b0000	Common not Private translations not supported.
0b0001	Common not Private translations supported.

All other values are reserved.

FEAT_TTCNP implements the functionality identified by the value 0b0001.

From Armv8.2, the only permitted value is 0b0001.

Accessing ID_AA64MMFR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b010

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_AA64MMFR2_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_AA64MMFR2_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_AA64MMFR2_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ID_AA64MMFR2_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = ID_AA64MMFR2_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ID_AA64MMFR3_EL1, AArch64 Memory Model Feature Register 3

The ID_AA64MMFR3_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

Configuration

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_AA64MMFR3_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Spec_FPACC				ADERR				SDERR				RES0				ANERR				SNERR				D128_2				D128			
MEC				AIE				S2POE				S1POE				S2PIE				S1PIE				SCTLRX				TCRX			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Spec_FPACC, bits [63:60] When FEAT_FPACCOMBINE is implemented:

Speculative behavior in the event of a PAC authentication failure in an implementation that includes FEAT_FPACCOMBINE. Defined values are:

Spec_FPACC	Meaning
0b0000	The implementation does not disclose whether the speculative use of pointers processed by a PAC Authentication is materially different in terms of the impact on cached microarchitectural state between passing and failing of the PAC Authentication.
0b0001	The speculative use of pointers processed by a PAC Authentication is not materially different in terms of the impact on cached microarchitectural state between passing and failing of the PAC Authentication.

All other values are reserved.

For the purpose of this definition, cached microarchitecture state is the state of caching agents such as instruction caches, data caches and TLBs which can be altered as a result of speculation caused by a mispredicted execution, but is not restored to the state prior to the speculation when the misprediction is corrected.

Otherwise:

Reserved, RES0.

ADERR, bits [59:56]

Asynchronous Device error exceptions. With ID_AA64MMFR3_EL1.SDERR, describes the PE behavior for error exceptions on Device memory loads.

ADERR	Meaning
0b0000	If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.SDERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.SDERR.
0b0001	Some error exceptions for Device memory loads are taken asynchronously.
0b0010	FEAT_ADERR is implemented. SCTLR2_ELx.EnADERR and HCRX_EL2.EnSDERR are implemented.

All other values are reserved.

When FEAT_RASv2 is implemented and ID_AA64MMFR3_EL1.SDERR is 0b0000, the value of this field is 0b0001.

When ID_AA64MMFR3_EL1.SDERR is 0b0001, the value of this field is 0b0000.

When ID_AA64MMFR3_EL1.SDERR is 0b0010, the value of this field is 0b0010.

FEAT_ADERR implements the functionality described by the value 0b0010.

SDERR, bits [55:52]

Synchronous Device error exceptions. With ID_AA64MMFR3_EL1.ADERR, describes the PE behavior for error exceptions on Device memory loads.

SDERR	Meaning
0b0000	If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.ADERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.ADERR.
0b0001	All error exceptions for Device memory loads are taken synchronously.
0b0010	FEAT_ADERR is implemented. SCTLR2_ELx.EnADERR and HCRX_EL2.EnSDERR are implemented.

All other values are reserved.

When FEAT_RASv2 is implemented and ID_AA64MMFR3_EL1.ADERR is 0b0000, the value of this field is 0b0001.

When ID_AA64MMFR3_EL1.ADERR is 0b0001, the value of this field is 0b0000.

When ID_AA64MMFR3_EL1.ADERR is 0b0010, the value of this field is 0b0010.

FEAT_ADERR implements the functionality described by the value 0b0010.

Bits [51:48]

Reserved, RES0.

ANERR, bits [47:44]

Asynchronous Normal error exceptions. With ID_AA64MMFR3_EL1.SNERR, describes the PE behavior for error exceptions on Normal memory loads.

ANERR	Meaning
0b0000	If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.SNERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.SNERR.
0b0001	Some error exceptions for Normal memory loads are taken asynchronously.
0b0010	FEAT_ANERR is implemented. SCTL2_ELx.EnANERR and HCRX_EL2.EnSNERR are implemented.

All other values are reserved.

When FEAT_RASv2 is implemented and ID_AA64MMFR3_EL1.SNERR is 0b0000, the value of this field is 0b0001.

When ID_AA64MMFR3_EL1.SNERR is 0b0001, the value of this field is 0b0000.

When ID_AA64MMFR3_EL1.SNERR is 0b0010, the value of this field is 0b0010.

FEAT_ANERR implements the functionality described by the value 0b0010.

SNERR, bits [43:40]

Synchronous Normal error exceptions. With ID_AA64MMFR3_EL1.ANERR, describes the PE behavior for error exceptions on Normal memory loads.

SNERR	Meaning
0b0000	If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.ANERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.ANERR.
0b0001	All error exceptions for Normal memory loads are taken synchronously.
0b0010	FEAT_ANERR is implemented. SCTL2_ELx.EnANERR and HCRX_EL2.EnSNERR are implemented.

All other values are reserved.

When FEAT_RASv2 is implemented and ID_AA64MMFR3_EL1.ANERR is 0b0000, the value of this field is 0b0001.

When ID_AA64MMFR3_EL1.ANERR is 0b0001, the value of this field is 0b0000.

When ID_AA64MMFR3_EL1.ANERR is 0b0010, the value of this field is 0b0010.

FEAT_ANERR implements the functionality described by the value 0b0010.

D128_2, bits [39:36]

128-bit Page Table Descriptor at stage 2. Indicates support for 128-bit Page Table Descriptor at stage 2. Defined values are:

D128_2	Meaning
0b0000	128-bit Page Table Descriptor Extension at stage 2 is not supported.
0b0001	128-bit Page Table Descriptor Extension at stage 2 is supported.

All other values are reserved.

D128, bits [35:32]

128-bit Page Table Descriptor. Indicates support for 128-bit Page Table Descriptor. Defined values are:

D128	Meaning
0b0000	128-bit Page Table Descriptor Extension is not supported.
0b0001	128-bit Page Table Descriptor Extension is supported.

All other values are reserved.

MEC, bits [31:28]

Indicates support for Memory Encryption Contexts. Defined values are:

MEC	Meaning
0b0000	Memory Encryption Contexts is not supported.
0b0001	Memory Encryption Contexts is supported for Realm physical address space.

All other values are reserved.

FEAT_MEC implements the functionality identified by the value 0b0001.

AIE, bits [27:24]

Attribute Indexing. Indicates support for the Attribute Index Extension. Defined values are:

AIE	Meaning
0b0000	The Attribute Index Extension is not supported.
0b0001	The Attribute Index Extension at Stage 1 is supported.

All other values are reserved.

S2POE, bits [23:20]

Stage 2 Permission Overlay. Indicates support for Permission Overlay at Stage 2. Defined values are:

S2POE	Meaning
0b0000	The Permission Overlay at Stage 2 is not supported.
0b0001	The Permission Overlay at Stage 2 is supported.

All other values are reserved.

S1POE, bits [19:16]

Stage 1 Permission Overlay. Indicates support for Permission Overlay at Stage 1. Defined values are:

S1POE	Meaning
0b0000	The Permission Overlay at Stage 1 is not supported.
0b0001	The Permission Overlay at Stage 1 is supported.

All other values are reserved.

S2PIE, bits [15:12]

Stage 2 Permission Indirection. Indicates support for Permission Indirection at Stage 2. Defined values are:

S2PIE	Meaning
0b0000	The Permission Indirection at Stage 2 is not supported.
0b0001	The Permission Indirection at Stage 2 is supported.

All other values are reserved.

S1PIE, bits [11:8]

Stage 1 Permission Indirection. Indicates support for Permission Indirection at Stage 1. Defined values are:

S1PIE	Meaning
0b0000	The Permission Indirection at Stage 1 is not supported.
0b0001	The Permission Indirection at Stage 1 is supported.

All other values are reserved.

SCTLRX, bits [7:4]

SCTLRX Extension. Indicates support for Extension of [SCTLR_EL1](#). Defined values are:

SCTLRX	Meaning
0b0000	SCTLR2_EL1 , SCTLR2_EL2 and their associated trap controls are not implemented.
0b0001	SCTLR2_EL1 , SCTLR2_EL2 and their associated trap controls are implemented.

All other values are reserved.

TCRX, bits [3:0]

TCR Extension. Indicates support for Extension of [TCR_EL1](#). Defined values are:

TCRX	Meaning
0b0000	TCR2_EL1 , TCR2_EL2 and their associated trap controls are not implemented.
0b0001	TCR2_EL1 , TCR2_EL2 and their associated trap controls are implemented.

All other values are reserved.

Accessing ID_AA64MMFR3_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64MMFR3_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b011

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_AA64MMFR3_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_AA64MMFR3_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR3_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR3_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR3_EL1;

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

ID_AA64MMFR4_EL1, AArch64 Memory Model Feature Register 4

The ID_AA64MMFR4_EL1 characteristics are:

Purpose

Provides additional information about implemented memory model and memory management support in AArch64.

Configuration

There are no configuration notes.

Attributes

ID_AA64MMFR4_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
																RES0																			
																				RES0								EIESB				RES0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:8]

Reserved, RES0.

EIESB, bits [7:4] When FEAT_IESB is implemented:

Indicates whether the implicit Error synchronization event inserted on taking an exception is inserted before or after the exception is taken.

EIESB	Meaning
0b0000	The behavior is not described.
0b0001	Implicit Error synchronization event is inserted before an exception is taken.
0b1111	Implicit Error synchronization event is inserted after an exception is taken.

All other values are reserved.

Implicit Error synchronization events are inserted on taking an exception to ELx when SCTLR_ELx.EISB is 0b1.

Inserting the event before the exception is taken means that if the Error synchronization event causes an SError exception to become pending, and SError exceptions are not masked and not disabled, then the SError exception is taken in place of the original exception.

Otherwise:

Reserved, RES0.

Bits [3:0]

Reserved, RES0.

Accessing ID_AA64MMFR4_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64MMFR4_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR4_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR4_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR4_EL1;
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

The ID_AA64PFR0_EL1 characteristics are:

Purpose

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

The external register [EDPFR](#) gives information from this register.

Attributes

ID_AA64PFR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
CSV3				CSV2				RME				DIT				AMU				MPAM			SEL2				SVE				
RAS				GIC				AdvSIMD				FP				EL3				EL2			EL1				ELO				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CSV3, bits [63:60]

Speculative use of faulting data. Defined values are:

CSV3	Meaning
0b0000	This PE does not disclose whether data loaded under speculation with a permission or domain fault can be used to form an address or generate condition codes or SVE predicate values to be used by other instructions in the speculative sequence.
0b0001	Data loaded under speculation with a permission or domain fault cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence. The execution timing of any other instructions in the speculative sequence is not a function of the data loaded under speculation.

All other values are reserved.

FEAT_CSV3 implements the functionality identified by the value 0b0001.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

If FEAT_E0PD is implemented, FEAT_CSV3 must be implemented.

CSV2, bits [59:56]

Speculative use of out of context branch targets. Defined values are:

CSV2	Meaning
0b0000	The implementation does not disclose whether FEAT_CSV2 is implemented.
0b0001	FEAT_CSV2 is implemented, but FEAT_CSV2_2 and FEAT_CSV2_3 are not implemented. ID_AA64PFR1_EL1 .CSV2_frac determines whether either or both of FEAT_CSV2_1p1 or FEAT_CSV2_1p2 are implemented.
0b0010	FEAT_CSV2_2 is implemented, but FEAT_CSV2_3 is not implemented.
0b0011	FEAT_CSV2_3 is implemented.

All other values are reserved.

FEAT_CSV2 implements the functionality identified by the value 0b0001.

FEAT_CSV2_2 implements the functionality identified by the value 0b0010.

FEAT_CSV2_3 implements the functionality identified by the feature 0b0011.

In Armv8.0, the permitted values are 0b0000, 0b0001, 0b0010, and 0b0011.

From Armv8.5, the permitted values are 0b0001, 0b0010, and 0b0011.

RME, bits [55:52]

Realm Management Extension (RME). Defined values are:

RME	Meaning
0b0000	Realm Management Extension not implemented.
0b0001	RMEv1 is implemented.

All other values are reserved.

FEAT_RME implements the functionality identified by the value 0b0001.

DIT, bits [51:48]

Data Independent Timing. Defined values are:

DIT	Meaning
0b0000	AArch64 does not guarantee constant execution time of any instructions.
0b0001	AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.

All other values are reserved.

FEAT_DIT implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

AMU, bits [47:44]

Indicates support for Activity Monitors Extension. Defined values are:

AMU	Meaning
0b0000	Activity Monitors Extension is not implemented.
0b0001	FEAT_AMUv1 is implemented.
0b0010	FEAT_AMUv1p1 is implemented. As 0b0001 and adds support for virtualization of the activity monitor event counters.

All other values are reserved.

FEAT_AMUv1 implements the functionality identified by the value 0b0001.

FEAT_AMUv1p1 implements the functionality identified by the value 0b0010.

In Armv8.0, the only permitted value is 0b0000.

In Armv8.4, the permitted values are 0b0000 and 0b0001.

From Armv8.6, the permitted values are 0b0000, 0b0001, and 0b0010.

MPAM, bits [43:40]

Indicates the major version number of support for the MPAM Extension.

Defined values are:

MPAM	Meaning
0b0000	The major version number of the MPAM extension is 0.
0b0001	The major version number of the MPAM extension is 1.

All other values are reserved.

When combined with the minor version number from [ID_AA64PFR1_EL1](#).MPAM_frac, the "major.minor" version is:

MPAM Extension version	MPAM	MPAM_frac
Not implemented.	0b0000	0b0000
v0.1 is implemented.	0b0000	0b0001
v1.0 is implemented.	0b0001	0b0000
v1.1 is implemented.	0b0001	0b0001

For more information, see 'The Memory Partitioning and Monitoring (MPAM) Extension'.

SEL2, bits [39:36]

Secure EL2. Defined values are:

SEL2	Meaning
0b0000	Secure EL2 is not implemented.
0b0001	Secure EL2 is implemented.

All other values are reserved.

FEAT_SEL2 implements the functionality identified by the value 0b0001.

SVE, bits [35:32]

Scalable Vector Extension. Defined values are:

SVE	Meaning
0b0000	SVE architectural state and programmers' model are not implemented.
0b0001	SVE architectural state and programmers' model are implemented.

All other values are reserved.

FEAT_SVE implements the functionality identified by the value 0b0001.

If implemented, refer to [ID_AA64ZFR0_EL1](#) for information about which SVE instructions are available.

RAS, bits [31:28]

RAS Extension version. **Defined values are:**

RAS	Meaning
0b0000	No RAS Extension.
0b0001	RAS Extension implemented.
0b0010	FEAT_RASv1p1 implemented and, if EL3 is implemented, FEAT_DoubleFault implemented. As 0b0001, and adds support for: <ul style="list-style-type: none"> If EL3 is implemented, FEAT_DoubleFault. Additional ERXMISC<m>_EL1 System registers. Additional System registers ERXPFPCDN_EL1, ERXPFPCCTL_EL1, and ERXPFPGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS and support for the optional RAS Timestamp and RAS Common Fault Injection Model Extensions.
0b0011	FEAT_RASv2 implemented. As 0b0010 and adds support for: <ul style="list-style-type: none"> ERXGSR_EL1, to support System RAS agents. Additional fine-grained EL2 traps for additional error record System registers. The SCR_EL3.ERR2En trap control for additional error record System registers. The SCR_EL3.TWERR write control for error record System registers. Error records accessed through System registers conform to RAS System Architecture v2.

All other values are reserved.

FEAT_RAS implements the functionality identified by the value 0b0001.

FEAT_RASv1p1 and FEAT_DoubleFault implement the functionality identified by the value 0b0010.

FEAT_RASv2 implements the functionality identified by the value 0b0011.

In Armv8.0 and Armv8.1, the permitted values are 0b0000 and 0b0001.

From Armv8.4, when FEAT_DoubleFault is not implemented, and [ERRIDR_EL1](#) is 0, the permitted values are IMPLEMENTATION DEFINED 0b0001 or 0b0010.

From In Armv8.2, the only permitted value is 0b0000 0b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented or [ERRIDR_EL1.NUM](#) is nonzero, the value 0b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented, the only permitted value is 0b0010.

Note

When the value of this field is 0b0001, [ID_AA64PFR1_EL1.RAS_frac](#) indicates whether FEAT_RASv1p1 is implemented.

GIC, bits [27:24]

System register GIC CPU interface. Defined values are:

GIC	Meaning
0b0000	GIC CPU interface system registers not implemented.
0b0001	System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.
0b0011	System register interface to version 4.1 of the GIC CPU interface is supported.

All other values are reserved.

AdvSIMD, bits [23:20]

Advanced SIMD. Defined values are:

AdvSIMD	Meaning
0b0000	Advanced SIMD is implemented, including support for the following Sisd and SIMD operations: <ul style="list-style-type: none"> Integer byte, halfword, word and doubleword element operations. Single-precision and double-precision floating-point arithmetic. Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.
0b0001	As for 0b0000, and also includes support for half-precision floating-point arithmetic.
0b1111	Advanced SIMD is not implemented.

All other values are reserved.

This field must have the same value as the FP field.

The permitted values are:

- 0b0000 in an implementation with Advanced SIMD support that does not include the FEAT_FP16 extension.
- 0b0001 in an implementation with Advanced SIMD support that includes the FEAT_FP16 extension.
- 0b1111 in an implementation without Advanced SIMD support.

FP, bits [19:16]

Floating-point. Defined values are:

FP	Meaning
0b0000	Floating-point is implemented, and includes support for: <ul style="list-style-type: none"> Single-precision and double-precision floating-point types. Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.
0b0001	As for 0b0000, and also includes support for half-precision floating-point arithmetic.
0b1111	Floating-point is not implemented.

All other values are reserved.

This field must have the same value as the AdvSIMD field.

The permitted values are:

- 0b0000 in an implementation with floating-point support that does not include the FEAT_FP16 extension.
- 0b0001 in an implementation with floating-point support that includes the FEAT_FP16 extension.
- 0b1111 in an implementation without floating-point support.

EL3, bits [15:12]

EL3 Exception level handling. Defined values are:

EL3	Meaning
0b0000	EL3 is not implemented.
0b0001	EL3 can be executed in AArch64 state only.
0b0010	EL3 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL2, bits [11:8]

EL2 Exception level handling. Defined values are:

EL2	Meaning
0b0000	EL2 is not implemented.
0b0001	EL2 can be executed in AArch64 state only.
0b0010	EL2 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL1, bits [7:4]

EL1 Exception level handling. Defined values are:

EL1	Meaning
0b0001	EL1 can be executed in AArch64 state only.
0b0010	EL1 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

EL0, bits [3:0]

EL0 Exception level handling. Defined values are:

EL0	Meaning
0b0001	EL0 can be executed in AArch64 state only.
0b0010	EL0 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

Accessing ID_AA64PFR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64PFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b000

```
if PSTATE.EL == EL0 then
  if IsFeatureImplemented(FEAT_IDST) then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      AArch64.SystemAccessTrap(EL1, 0x18);
  else
    UNDEFINED;
elsif PSTATE.EL == EL1 then
  if EL2Enabled() && HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
  X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
  X[t, 64] = ID_AA64PFR0_EL1;
```

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

The ID_AA64PFR1_EL1 characteristics are:

Purpose

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64PFR1_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
PFARRES0				RES0NMI				MTEXCSV2_frac				THE				RES0				MTE_frac				NMI				CSV2_frac			
RNDR_trap				SME				RES0				MPAM_frac				RAS_frac				MTE				SSBS				BT			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PFAR, Bits bits [63:6040]

Support for physical fault address registers, FEAT_PFAR.

PFAR	Meaning
0b0000	FEAT_PFAR is not implemented.
0b0001	FEAT_PFAR is implemented. Includes support for the PFAR ELx and, if EL3 is implemented, MFAR_EL3 registers.

All other values are reserved.

FEAT_PFAR implements the functionality identified by the value 0b0001.

Bits [59:56]

Reserved, RES0.

MTEX, bits [55:52]

Additional tag checking modes for MTE. Defined values are:

MTEX	Meaning
0b0000	Support for Memory Tagging when Address tagging is enabled.
0b0001	As 0b0000, and the following additional modes are supported: <ul style="list-style-type: none"> Canonical Tag checking, identified as FEAT_MTE_CANONICAL_TAGS. Memory tagging with Address tagging disabled, identified as FEAT_MTE_NO_ADDRESS_TAGS.

Note

ID_AA64PFR1_EL1.MTE identifies Memory Tagging when Address tagging is enabled.

ID_AA64PFR1_EL1.MTE_frac identifies whether or not Asynchronous Faulting and asymmetric Tag Check Fault handling are supported, in conjunction with the value of ID_AA64PFR1_EL1.MTE. Support for Asynchronous Faulting and asymmetric Tag Check Fault handling applies to all tag checking modes, including those introduced by ID_AA64PFR1_EL1.MTEX.

All other values are reserved.

This field is valid only if ID_AA64PFR1_EL1.MTE ≥ 0b0010.

FEAT_MTE4 implements the functionality identified by the value 0b0001.

From Armv8.9, the only permitted value is 0b0001.

THE, bits [51:48]

Support for Translation Hardening Extension. Defined values are:

THE	Meaning
0b0000	Translation Hardening Extension is not implemented.
0b0001	The RCW and RCWS instructions, their associated registers and traps are supported. If EL2 is implemented, the AssuredOnly check, TopLevel check, and their associated controls are also implemented.

All other values are reserved.

FEAT_THE implements the functionality identified by the value 0b0001.

Bits [47:44]

Reserved, RES0.

MTE_frac, bits [43:40]

Support for Asynchronous Faulting and asymmetric Tag Check Fault handling. Defined values are:

MTE_frac	Meaning
0b0000	Asynchronous Faulting is supported. If ID_AA64PFR1_EL1.MTE ≥ 0b0011, asymmetric Tag Check Fault handling is supported.
0b1111	Asynchronous Faulting is not supported. If ID_AA64PFR1_EL1.MTE ≥ 0b0011, asymmetric Tag Check Fault handling is not supported.

All other values are reserved.

This field is valid only if ID_AA64PFR1_EL1.MTE ≥ 0b0010.

NMI, bits [39:36]

Non-maskable Interrupt. Indicates support for Non-maskable interrupts. Defined values are:

NMI	Meaning
0b0000	SCTLR_ELx.{SPINTMASK, NMI} and PSTATE.ALLINT with its associated instructions are not supported.
0b0001	SCTLR_ELx.{SPINTMASK, NMI} and PSTATE.ALLINT with its associated instructions are supported.

All other values are reserved.

FEAT_NMI implements the functionality identified by the value 0b0001.

From Armv8.8, the only permitted value is 0b0001.

CSV2_frac, bits [35:32]

CSV2 fractional field. Defined values are:

CSV2_frac	Meaning
0b0000	Either ID_AA64PFR0_EL1 .CSV2 is not 0b0001, or the implementation does not disclose whether FEAT_CSV2_1p1 is implemented. FEAT_CSV2_1p2 is not implemented.
0b0001	FEAT_CSV2_1p1 is implemented, but FEAT_CSV2_1p2 is not implemented.
0b0010	FEAT_CSV2_1p2 is implemented.

All other values are reserved.

FEAT_CSV2_1p1 implements the functionality identified by the value 0b0001.

FEAT_CSV2_1p2 implements the functionality identified by the value 0b0010.

From Armv8.0, the permitted values are 0b0000, 0b0001, and 0b0010.

The values 0b0001 and 0b0010 are permitted only when [ID_AA64PFR0_EL1](#).CSV2 is 0b0001.

RNDR_trap, bits [31:28]

Random Number trap to EL3 field. Defined values are:

RNDR_trap	Meaning
0b0000	Trapping of RNDR and RNDRRS to EL3 is not supported.
0b0001	Trapping of RNDR and RNDRRS to EL3 is supported. SCR_EL3 .TRNDR is present.

All other values are reserved.

FEAT_RNG_TRAP implements the functionality identified by the value 0b0001.

SME, bits [27:24]

Scalable Matrix Extension. Defined values are:

SME	Meaning
0b0000	SME architectural state and programmers' model are not implemented.
0b0001	SME architectural state and programmers' model are implemented.
0b0010	As 0b0001, plus the SME2 ZT0 register.

All other values are reserved.

FEAT_SME implements the functionality identified by the value 0b0001.

FEAT_SME2 implements the functionality identified by the value 0b0010.

From Armv9.2, the permitted values are 0b0000 and 0b0001.

From Armv9.2, the permitted values are 0b0000, 0b0001, and 0b0010.

If implemented, refer to [ID_AA64SMFR0_EL1](#) and [ID_AA64ZFR0_EL1](#) for information about which SME and SVE instructions are available.

Bits [23:20]

Reserved, RES0.

MPAM_frac, bits [19:16]

Indicates the minor version number of support for the MPAM Extension.

Defined values are:

MPAM_frac	Meaning
0b0000	The minor version number of the MPAM extension is 0.
0b0001	The minor version number of the MPAM extension is 1.

All other values are reserved.

When combined with the major version number from [ID_AA64PFR0_EL1](#).MPAM, The combined "major.minor" version is:

MPAM Extension version	MPAM	MPAM_frac
Not implemented.	0b0000	0b0000
v0.1 is implemented.	0b0000	0b0001
v1.0 is implemented.	0b0001	0b0000
v1.1 is implemented.	0b0001	0b0001

For more information, see 'The Memory Partitioning and Monitoring (MPAM) Extension'.

RAS_frac, bits [15:12]

RAS Extension fractional field. Defined values are:

RAS_frac	Meaning
0b0000	If ID_AA64PFR0_EL1 .RAS == 0b0001, RAS Extension implemented.
0b0001	If ID_AA64PFR0_EL1 .RAS == 0b0001, as 0b0000 and adds support for: <ul style="list-style-type: none"> Additional ERXMISC<m>_EL1 System registers. Additional System registers ERXPGCDN_EL1, ERXPGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS , and support for the optional RAS Timestamp and RAS Common Fault Injection Model Extensions.

All other values are reserved.

FEAT_RASv1p1 implements the functionality identified by the value 0b0001.

This field is valid only if [ID_AA64PFR0_EL1](#).RAS == 0b0001.

MTE, bits [11:8]

Support for the Memory Tagging Extension. Defined values are:

MTE	Meaning
0b0000	Memory Tagging Extension is not implemented.
0b0001	Instruction-only Memory Tagging Extension is implemented.
0b0010	Full Memory Tagging Extension is implemented. Support for Asynchronous Faulting is defined by ID_AA64PFR1_EL1.MTE_frac
0b0011	Memory Tagging Extension is implemented with support for asymmetric Tag Check Fault handling. Support for Asynchronous Faulting and asymmetric Tag Check Fault handling is defined by ID_AA64PFR1_EL1.MTE_frac.

All other values are reserved.

FEAT_MTE implements the functionality identified by the value 0b0001.

FEAT_MTE2 implements the functionality identified by the value 0b0010.

FEAT_MTE3 implements the functionality identified by the value 0b0011.

In Armv8.5, the permitted values are 0b0000, 0b0001, 0b0010, and 0b0011.

From Armv8.7, the value 0b0010 is not permitted.

SSBS, bits [7:4]

Speculative Store Bypassing controls in AArch64 state. Defined values are:

SSBS	Meaning
0b0000	AArch64 provides no mechanism to control the use of Speculative Store Bypassing.
0b0001	AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypass Safe.
0b0010	As 0b0001, and adds the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.

All other values are reserved.

FEAT_SSBS implements the functionality identified by the value 0b0001.

FEAT_SSBS2 implements the functionality identified by the value 0b0010.

BT, bits [3:0]

Branch Target Identification mechanism support in AArch64 state. Defined values are:

BT	Meaning
0b0000	The Branch Target Identification mechanism is not implemented.
0b0001	The Branch Target Identification mechanism is implemented.

All other values are reserved.

FEAT_BTI implements the functionality identified by the value 0b0001.

From Armv8.5, the only permitted value is 0b0001.

Accessing ID_AA64PFR1_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64PFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b001

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_AA64PFR1_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = ID_AA64PFR1_EL1;
    elseif PSTATE.EL == EL3 then
        X[t, 64] = ID_AA64PFR1_EL1;

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

ID_AA64PFR2_EL1, AArch64 Processor Feature Register 2

The ID_AA64PFR2_EL1 characteristics are:

Purpose

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64PFR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																MTEFAR				MTESTOREONLY				MTEPERM							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:12]

Reserved, RES0.

MTEFAR, bits [11:8]

Information reported in FAR_ELx on a synchronous exception due to a Tag Check Fault. Defined values are:

MTEFAR	Meaning
0b0000	On a synchronous exception due to a Tag Check Fault, FAR_ELx [63:60] is UNKNOWN. Support for Memory Tagging when Address tagging is enabled.
0b0001	On a synchronous exception due to a Tag Check Fault, FAR_ELx [63:60] is not UNKNOWN. This feature is identified as FEAT_MTE_TAGGED_FAR.

All other values are reserved.

This field is valid when [ID_AA64PFR1_EL1](#).MTE >= 0b0010 and [ID_AA64PFR1_EL1](#).MTEx >= 0b0001.

FEAT_MTE4 implements the functionality identified by the value 0b0001.

From Armv8.9, the only permitted value is 0b0001.

MTESTOREONLY, bits [7:4]

Store-only Tag checking, identified as FEAT_MTE_STORE_ONLY. Defined values are:

MTESTOREONLY	Meaning
0b0000	FEAT_MTE_STORE_ONLY is not supported.
0b0001	FEAT_MTE_STORE_ONLY is supported.

All other values are reserved.

This field is valid when [ID_AA64PFR1_EL1.MTE](#) \geq 0b0010 and [ID_AA64PFR1_EL1.MTEX](#) \geq 0b0001.

FEAT_MTE4 implements the functionality identified by the value 0b0001.

From Armv8.9, the only permitted value is 0b0001.

MTEPERM, bits [3:0]

Tag access permissions. Defined values are:

MTEPERM	Meaning
0b0000	FEAT_MTE_PERM is not supported.
0b0001	FEAT_MTE_PERM is supported.

All other values are reserved.

This field is valid when [ID_AA64PFR1_EL1.MTE](#) \geq 0b0010.

Accessing ID_AA64PFR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64PFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b010

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_AA64PFR2_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_AA64PFR2_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR2_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR2_EL1;

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SMEver	Meaning
0b0000	The mandatory SME instructions are implemented.
0b0001	As 0b0000, and adds the mandatory SME2 instructions.
0b0010	As 0b0001, and adds the mandatory SME2.1 instructions.

All other values are reserved.

If FEAT_SME is implemented and FEAT_SME2 is not implemented, the only permitted value is 0b0000.

If FEAT_SME2 is implemented and FEAT_SME2p1 is not implemented, the only permitted value is 0b0001.

FEAT_SME2p1 implements the functionality identified by 0b0010.

Otherwise:

Reserved, RES0.

I16I64, bits [55:52]

Indicates SME support for instructions that accumulate into 64-bit integer elements in the ZA array. Defined values are:

I16I64	Meaning
0b0000	Instructions that accumulate into 64-bit integer elements in the ZA array are not implemented.
0b1111	The variants of the ADDHA, ADDVA, SMOPA, SMOPS, SUMOPA, SUMOPS, UMOA, UMOPS, USMOPA, and USMOPS instructions that accumulate into 64-bit integer tiles are implemented. When FEAT_SME2 is implemented, the variants of the ADD, ADDA, SDOT, SMLALL, SMLSLL, SUB, SUBA, SVDOT, UDOT, UMLALL, UMLSLL, and UVDOT instructions that accumulate into 64-bit integer elements in ZA array vectors are implemented.

All other values are reserved.

FEAT_SME_I16I64 implements the functionality identified by the value 0b1111.

The only permitted values are 0b0000 and 0b1111.

Bits [51:49]

Reserved, RES0.

F64F64, bit [48]

Indicates SME support for instructions that accumulate into FP64 double-precision floating-point elements in the ZA array. Defined values are:

F64F64	Meaning
0b0	Instructions that accumulate into double-precision floating-point elements in the ZA array are not implemented.
0b1	The variants of the FMOPA and FMOPS instructions that accumulate into double-precision tiles are implemented. When FEAT_SME2 is implemented, the variants of the FADD, FMLA, FMLS, and FSUB instructions that accumulate into double-precision elements in ZA array vectors are implemented.

FEAT_SME_F64F64 implements the functionality identified by the value 0b1.

I16I32, Bits bits [47:4440]

Indicates SME2 support for instructions that accumulate 16-bit outer products into 32-bit integer tiles. Defined values are:

I16I32	Meaning
0b0000	Instructions that accumulate 16-bit outer products into 32-bit integer tiles are not implemented.
0b0101	The SMOPA (2-way), SMOPS (2-way), UMOPA (2-way), and UMOPS (2-way) instructions that accumulate 16-bit outer products into 32-bit integer tiles are implemented.

All other values are reserved.

If FEAT_SME2 is implemented, the only permitted value is 0b0101. Otherwise, the only permitted value is 0b0000.

B16B16, bit [43]

Indicates support for SME2.1 non-widening BFloat16 instructions. Defined values are:

B16B16	Meaning
0b0	SME2.1 non-widening BFloat16 instructions are not implemented.
0b1	SME2.1 non-widening BFloat16 instructions are implemented.

FEAT_B16B16 implements the functionality identified by 0b1.

This field must indicate the same level of support as [ID_AA64ZFR0_EL1.B16B16](#).

If one or more of FEAT_SVE2p1 and FEAT_SME2p1 is implemented, the values 0b0 and 0b1 are permitted.

Otherwise, the only permitted value is 0b0.

F16F16, bit [42]

Indicates support for SME2.1 non-widening half-precision floating-point instructions. Defined values are:

F16F16	Meaning
0b0	SME2.1 non-widening half-precision floating-point instructions are not implemented.
0b1	SME2.1 non-widening half-precision floating-point instructions are implemented.

FEAT_SME_F16F16 implements the functionality identified by 0b1.

If FEAT_SME2p1 is implemented, the values 0b0 and 0b1 are permitted.

Otherwise, the only permitted value is 0b0.

Bits [41:40]

Reserved, RES0.

I8I32, bits [39:36]

Indicates SME support for instructions that accumulate 8-bit integer outer products into 32-bit integer tiles. Defined values are:

I8I32	Meaning
0b0000	Instructions that accumulate 8-bit outer products into 32-bit tiles are not implemented.
0b1111	The SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate 8-bit outer products into 32-bit tiles are implemented.

All other values are reserved.

If FEAT_SME is implemented, the only permitted value is 0b1111.

F16F32, bit [35]

Indicates SME support for instructions that accumulate FP16 half-precision floating-point outer products into FP32 single-precision floating-point tiles. Defined values are:

F16F32	Meaning
0b0	Instructions that accumulate half-precision outer products into single-precision tiles are not implemented.
0b1	The FMOPA and FMOPS instructions that accumulate half-precision outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is 0b1.

B16F32, bit [34]

Indicates SME support for instructions that accumulate BFloat16 outer products into FP32 single-precision floating-point tiles. Defined values are:

B16F32	Meaning
0b0	Instructions that accumulate BFloat16 outer products into single-precision tiles are not implemented.
0b1	The BFMOPA and BFMOPS instructions that accumulate BFloat16 outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is 0b1.

BI32I32, bit [33]

Indicates SME support for instructions that accumulate thirty-two 1-bit binary outer products into 32-bit integer tiles. Defined values are:

Reserved, RES0.

BI32I32	Meaning
0b0	Instructions that accumulate 1-bit binary outer products into 32-bit integer tiles are not implemented.
0b1	The BMOPA and BMOPS instructions that accumulate 1-bit binary outer products into 32-bit integer tiles are implemented.

If FEAT_SME2 is implemented, the only permitted value is 0b1. Otherwise, the only permitted value is 0b0.

F32F32, bit [32]

Indicates SME support for instructions that accumulate FP32 single-precision floating-point outer products into single-precision floating-point tiles. Defined values are:

F32F32	Meaning
0b0	Instructions that accumulate single-precision outer products into single-precision tiles are not implemented.
0b1	The FMOPA and FMOPS instructions that accumulate single-precision outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is 0b1.

Bits [31:0]

Reserved, RES0.

ID_AA64ZFR0_EL1, SVE Feature ID register 0

The ID_AA64ZFR0_EL1 characteristics are:

Purpose

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension instruction set, when one or more of FEAT_SVE and FEAT_SME is implemented.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

If FEAT_SME is implemented and FEAT_SVE is not implemented, then SVE instructions can only be executed when the PE is in Streaming SVE mode and the instructions are legal to execute in Streaming SVE mode.

Attributes

ID_AA64ZFR0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0	F64MM	F32MM	RES0	I8MM	SM4	RES0	SHA3	RES0	B16B16BF16	BF16BitPerm	BitPermRES0	RES0AES	AESSVEEver	SVEEver																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:60]

Reserved, RES0.

F64MM, bits [59:56]

Indicates support for SVE FP64 double-precision floating-point matrix multiplication instructions. Defined values are:

F64MM	Meaning
0b0000	Double-precision matrix multiplication and related SVE instructions are not implemented.
0b0001	Double-precision variant of the FMMLA instruction, and the LD1RO* instructions are implemented. The 128-bit element variantsvariations of the SVE TRN1, TRN2, UZP1, UZP2, ZIP1, and ZIP2 instructions are also implemented.

All other values are reserved.

FEAT_F64MM implements the functionality identified by 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

F32MM, bits [55:52]

Indicates support for the SVE FP32 single-precision floating-point matrix multiplication instruction. Defined values are:

F32MM	Meaning
0b0000	Single-precision matrix multiplication instruction is not implemented.
0b0001	Single-precision variant of the FMMLA instruction is implemented.

All other values are reserved.

FEAT_F32MM implements the functionality identified by 0b0001.

From Arm v8.2, the permitted values are 0b0000 and 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

Bits [51:48]

Reserved, RES0.

I8MM, bits [47:44]

Indicates support for SVE Int8 matrix multiplication instructions. Defined values are:

I8MM	Meaning
0b0000	SVE Int8 matrix multiplication instructions are not implemented.
0b0001	SVE SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.

All other values are reserved.

FEAT_I8MM implements the functionality identified by 0b0001.

When Advanced SIMD and SVE are both implemented, this field must return the same value as [ID_AA64ISAR1_EL1](#).I8MM.

From Armv8.6, the only permitted value is 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the SVE instructions SMMLA, UMMLA, and USMMLA, irrespective of the value of this field.

SM4, bits [43:40]

Indicates support for SVE SM4 instructions. Defined values are:

SM4	Meaning
0b0000	SVE SM4 instructions are not implemented.
0b0001	SVE SM4E and SM4EKEY instructions are implemented.

All other values are reserved.

FEAT_SVE_SM4 implements the functionality identified by 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

Bits [39:36]

Reserved, RES0.

SHA3, bits [35:32]

Indicates support for the SVE SHA3 instructions. Defined values are:

SHA3	Meaning
0b0000	SVE SHA3 instructions are not implemented.
0b0001	SVE RAX1 instruction is implemented.

All other values are reserved.

FEAT_SVE_SHA3 implements the functionality identified by 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

However, if both FEAT_SME2p1 and FEAT_SVE_SHA3 are implemented, then the SVE RAX1 instruction can be executed when the PE is in Streaming SVE mode regardless of whether FEAT_SME_FA64 is implemented and enabled at the current Exception level.

Bits [31:28:24]

Reserved, RES0.

B16B16, bits [27:24]

Indicates support for SVE2.1 non-widening BFloat16 instructions. Defined values are:

B16B16	Meaning
0b0000	SVE2.1 non-widening BFloat16 instructions are not implemented.
0b0001	SVE2.1 non-widening BFloat16 instructions are implemented.

FEAT_B16B16 implements the functionality identified by 0b0001.

This field must indicate the same level of support as [ID_AA64SMFR0_EL1.B16B16](#).

If one or more of FEAT_SVE2p1 and FEAT_SME2p1 is implemented, the values 0b0000 and 0b0001 are permitted.

Otherwise, the only permitted value is 0b0000.

BF16, bits [23:20]

Indicates support for SVE BFloat16 instructions. Defined values are:

BF16	Meaning
0b0000	SVE BFloat16 instructions are not implemented.
0b0001	SVE BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.
0b0010	As 0b0001, but the EPCR.EBF field is also supported.

All other values are reserved.

FEAT_BF16 **addsimplements** the functionality identified by 0b0001.

FEAT_EBF16 **addsimplements** the functionality identified by 0b0010.

This field must return the same value as [ID_AA64ISAR1_EL1](#).BF16.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the SVE instruction BFMMLA, irrespective of the value of this field.

If FEAT_SME is implemented, the permitted values are 0b0001 and 0b0010.

From ~~Otherwise, from~~ Armv8.6 and Armv9.1, the ~~only permitted~~ value ~~is~~ 0b0000 ~~0b0001~~ is not permitted..

BitPerm, bits [19:16]

Indicates support for SVE bit permute instructions. Defined values are:

BitPerm	Meaning
0b0000	SVE bit permute instructions are not implemented.
0b0001	SVE BDEP, BEXT, and BGRP instructions are implemented.

All other values are reserved.

FEAT_SVE_BitPerm implements the functionality identified by 0b0001.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

Bits [15:8]

Reserved, RES0.

AES, bits [7:4]

Indicates support for SVE AES instructions. Defined values are:

AES	Meaning
0b0000	SVE AES* instructions are not implemented.
0b0001	SVE AESE, AESD, AESMC, and AESIMC instructions are implemented.
0b0010	As 0b0001, plus 64-bit source element variants of SVE PMULLB and PMULLT instructions are with implemented. 64-bit source.

All other values are reserved.

FEAT_SVE_AES implements the functionality identified by the value 0b0001.

FEAT_SVE_PMULL128 implements the functionality identified by the value 0b0010.

The permitted values are 0b0000 and 0b0010.

When the PE is in Streaming SVE mode and it is not known whether FEAT_SME_FA64 is implemented and enabled at the current Exception level, software should not attempt to execute the instructions described by non-zero values of this field, irrespective of the value of this field.

SVEver, bits [3:0]

Indicates support for SVE instructions when one or more of FEAT_SME and FEAT_SVE is implemented. Defined values are:

SVEver	Meaning
0b0000	The SVE instructions are implemented.
0b0001	As 0b0000, and adds the mandatory SVE2 instructions.
0b0010	As 0b0001, and adds the mandatory SVE2.1 instructions.

All other values are reserved.

From Armv9, if this register is present, the value 0b0000 is not permitted.

FEAT_SVE2 implements the functionality identified by 0b0001 when the PE is not in Streaming SVE mode.

FEAT_SME implements the functionality identified by 0b0001 when the PE is in Streaming SVE mode.

FEAT_SME2p1 implements the functionality identified by 0b0010 when the PE is in Streaming SVE mode.

FEAT_SVE2p1 implements the functionality identified by 0b0010 when the PE is not in Streaming SVE mode.

From Armv9.4, the value 0b0001 is not permitted.

Accessing ID_AA64ZFR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_AA64ZFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b100

```
if PSTATE.EL == EL0 then
  if IsFeatureImplemented(FEAT_IDST) then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      AArch64.SystemAccessTrap(EL1, 0x18);
  else
    UNDEFINED;
elsif PSTATE.EL == EL1 then
  if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_AA64ZFR0_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_AA64ZFR0_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
  X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
  X[t, 64] = ID_AA64ZFR0_EL1;
```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ID_DFR0_EL1, AArch32 Debug Feature Register 0

The ID_DFR0_EL1 characteristics are:

Purpose

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch64 System register ID_DFR0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ID_DFR0\[31:0\]](#).

Attributes

ID_DFR0_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
TraceFilt				PerfMon				MProfDbg				MMapTrc				CopTrc				MMapDbg				CopSDBG				CopDbg			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

TraceFilt, bits [31:28]

Armv8.4 Self-hosted Trace Extension version. Defined values are:

TraceFilt	Meaning
0b0000	Armv8.4 Self-hosted Trace Extension not implemented.
0b0001	Armv8.4 Self-hosted Trace Extension implemented.

All other values are reserved.

FEAT_TRF implements the functionality added by the value 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

PerfMon, bits [27:24]

Performance Monitors Extension version.

This field does not follow the standard ID scheme, but uses the alternative ID scheme described in 'Alternative ID scheme used for the Performance Monitors Extension version'

Defined values are:

PerfMon	Meaning
0b0000	Performance Monitors Extension not implemented.
0b0001	Performance Monitors Extension, PMUv1 implemented.
0b0010	Performance Monitors Extension, PMUv2 implemented.
0b0011	Performance Monitors Extension, PMUv3 implemented.
0b0100	PMUv3 for Armv8.1. As 0b0011, and adds support for: <ul style="list-style-type: none"> Extended 16-bit PMEVTYPER<n>.evtCount field. If EL2 is implemented, the HDCR.HPMD control.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds support for the PMMIR register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the HDCR.HCCD control. If EL3 is implemented, the MDCR_EL3.SCCD control.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds support for: <ul style="list-style-type: none"> The PMCR.FZO and, if EL2 is implemented, HDCR.HPMFZO controls. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls.
0b1000	PMUv3 for Armv8.8. As 0b0111, and: <ul style="list-style-type: none"> Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. Removes the CONSTRAINED UNPREDICTABLE behaviors if a reserved or unimplemented PMU event number is selected.
0b1001	PMUv3 for Armv8.9. As 0b1000, and: <ul style="list-style-type: none"> Updates the definitions of existing PMU events. Adds support for the EDECR.PME control.
0b1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. Arm does not recommend this value for new implementations.

All other values are reserved.

FEAT_PMUv3 implements the functionality identified by the value 0b0011.

FEAT_PMUv3p1 implements the functionality identified by the value 0b0100.

FEAT_PMUv3p4 implements the functionality identified by the value 0b0101.

FEAT_PMUv3p5 implements the functionality identified by the value 0b0110.

FEAT_PMUv3p7 implements the functionality identified by the value 0b0111.

FEAT_PMUv3p8 implements the functionality identified by the value 0b1000.

FEAT_PMUv3p9 implements the functionality identified by the value 0b1001.

In any Armv8 implementation, the values 0b0001 and 0b0010 are not permitted.

From Armv8.1, if FEAT_PMUv3 is implemented, the value 0b0011 is not permitted.

From Armv8.4, if FEAT_PMUv3 is implemented, the value 0b0100 is not permitted.

From Armv8.5, if FEAT_PMUv3 is implemented, the value 0b0101 is not permitted.

From Armv8.7, if FEAT_PMUv3 is implemented, the value 0b0110 is not permitted.

From Armv8.8, if FEAT_PMUv3 is implemented, the value 0b0111 is not permitted.

Note

PMUv1 and PMUv2 are not permitted in an Armv8 implementation.

From Armv8.9, if FEAT_PMUv3 is implemented, the value 0b1000 is not permitted.

MProfDbg, bits [23:20]

M-profile Debug. Support for memory-mapped debug model for M-profile processors. Defined values are:

MProfDbg	Meaning
0b0000	Not supported.
0b0001	Support for M-profile Debug architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

MMapTrc, bits [19:16]

Memory-mapped Trace. Support for memory-mapped trace model. Defined values are:

MMapTrc	Meaning
0b0000	Not supported.
0b0001	Support for Arm trace architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the permitted values are 0b0000 and 0b0001.

For more information, see the ARM® Embedded Trace Macrocell Architecture Specification, ETMv4 (ARM IHI 0064).

CopTrc, bits [15:12]

Support for System registers-based trace model, using registers in the coproc == 0b1110 encoding space. Defined values are:

CopTrc	Meaning
0b0000	Not supported.
0b0001	Support for Arm trace architecture, with System registers access.

All other values are reserved.

In Armv8-A, the permitted values are 0b0000 and 0b0001.

For more information, see the ARM® Embedded Trace Macrocell Architecture Specification, ETMv4 (ARM IHI 0064).

MMapDbg, bits [11:8]

Memory-mapped Debug. Support for Armv7 memory-mapped debug model for A and R-profile processors. Defined values are:

MMapDbg	Meaning
0b0000	Not supported.
0b0100	Support for Armv7, v7 Debug architecture, with memory-mapped access.
0b0101	Support for Armv7, v7.1 Debug architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

The optional memory map defined by Armv8 is not compatible with Armv7.

CopSDBG, bits [7:4]

Support for a System registers-based Secure debug model, using registers in the coproc = 0b1110 encoding space, for an A-profile processor that includes EL3.

If EL3 is not implemented and the implemented Security state is Non-secure state, this field is RES0. Otherwise, this field reads the same as bits [3:0].

CopDBG, bits [3:0]

Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are:

CopDBG	Meaning
0b0000	Not supported.
0b0010	Armv6, v6 Debug architecture, with System registers access.
0b0011	Armv6, v6.1 Debug architecture, with System registers access.
0b0100	Armv7, v7 Debug architecture, with System registers access.
0b0101	Armv7, v7.1 Debug architecture, with System registers access.
0b0110	Armv8 debug architecture.
0b0111	Armv8 debug architecture with Virtualization Host Extensions.
0b1000	Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.
0b1011	Armv8.9 debug architecture, FEAT_Debugv8p9.

All other values are reserved.

The values 0b0000, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8.

FEAT_VHE adds the functionality identified by the value 0b0111.

FEAT_Debugv8p2 adds the functionality identified by the value 0b1000.

FEAT_Debugv8p4 adds the functionality identified by the value 0b1001.

FEAT_Debugv8p8 adds the functionality identified by the value 0b1010.

FEAT_Debugv8p9 adds the functionality identified by the value 0b1011.

From Armv8.1, when FEAT_VHE is implemented the value 0b0110 is not permitted.

From Armv8.2, the values 0b0110 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 is not permitted.

From Armv8.9, the value 0b1010 is not permitted.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
UNKNOWN																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Reserved, UNKNOWN.

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b010

(old)

htmldiff from-

(new)

ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6

The ID_ISAR6_EL1 characteristics are:

Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID_ISAR0_EL1](#), [ID_ISAR1_EL1](#), [ID_ISAR2_EL1](#), [ID_ISAR3_EL1](#), [ID_ISAR4_EL1](#) and [ID_ISAR5_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch64 System register ID_ISAR6_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ID_ISAR6\[31:0\]](#).

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_ISAR6_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
CLRBHBRES0				I8MM				BF16				SPECRES				SB				FHM				DP				JSCVT			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32:28]

Reserved, RES0.

CLRBHB, bits [31:28]

Indicates support for the CLRBHB instruction in AArch32 state. Defined values are:

CLRBHB	Meaning
0b0000	CLRBHB instruction is not implemented.
0b0001	CLRBHB instruction is implemented.

All other values are reserved.

FEAT_CLRBHB implements the functionality identified by 0b0001.

From Armv8.9, the value 0b0000 is not permitted.

I8MM, bits [27:24]

Indicates support for Advanced SIMD and floating-point Int8 matrix multiplication instructions in AArch32 state. Defined values of this field are:

I8MM	Meaning
0b0000	Int8 matrix multiplication instructions are not implemented.
0b0001	VSMMLA, VSUDOT, VUMMLA, VUSMMLA, and VUSDOT instructions are implemented.

All other values are reserved.

FEAT_AA32I8MM implements the functionality identified by 0b0001.

BF16, bits [23:20]

Indicates support for Advanced SIMD and floating-point BFloat16 instructions in AArch32 state. Defined values are:

BF16	Meaning
0b0000	BFloat16 instructions are not implemented.
0b0001	VCVT, VCVTB, VCVTT, VDOT, VFMA, VFMA, and VMMLA instructions with BF16 operand or result types are implemented.

All other values are reserved.

FEAT_AA32BF16 implements the functionality identified by 0b0001.

SPECRES, bits [19:16]

Indicates support for ~~prediction~~~~Speculation~~ invalidation instructions in AArch32 state. Defined values are:

SPECRES	Meaning
0b0000	Prediction invalidation instructions are not implemented.
0b0001	CFPRCTX , DVPRCTX , and CPRCTX instructions are implemented.
0b0010	As 0b0001, and COSPRCTX instruction is implemented.

All other values are reserved.

FEAT_SPECRES implements the functionality identified by 0b0001.

FEAT_SPECRES2 ~~From~~ implements Armv8.5, the ~~functionality only~~ ~~identified~~ ~~permitted~~ ~~by~~ ~~value is~~ 0b0010 ~~0b0001~~.

From Armv8.5, the value 0b0000 is not permitted.

From Armv8.9, the value 0b0001 is not permitted.

SB, bits [15:12]

Indicates support for the SB instruction in AArch32 state. Defined values are:

SB	Meaning
0b0000	SB instruction is not implemented.
0b0001	SB instruction is implemented.

All other values are reserved.

FEAT_SB implements the functionality identified by 0b0001.

From Armv8.5, the only permitted value is 0b0001.

FHM, bits [11:8]

Indicates support for Advanced SIMD and floating-point VFMA and VFMSL instructions in AArch32 state. Defined values are:

FHM	Meaning
0b0000	VFMA and VFMSL instructions are not implemented.
0b0001	VFMA and VFMSL instructions are implemented.

All other values are reserved.

FEAT_FHM implements the functionality identified by 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

DP, bits [7:4]

Indicates support for dot product instructions in AArch32 state. Defined values are:

DP	Meaning
0b0000	Dot product instructions are not implemented.
0b0001	VUDOT and VSDOT instructions are implemented.

All other values are reserved.

FEAT_DotProd implements the functionality identified by 0b0001.

In Armv8.2, the permitted values are 0b0000 and 0b0001.

From Armv8.4, the only permitted value is 0b0001.

JSCVT, bits [3:0]

Indicates support for the VJCVT instruction in AArch32 state. Defined values are:

JSCVT	Meaning
0b0000	The VJCVT instruction is not implemented.
0b0001	The VJCVT instruction is implemented.

All other values are reserved.

FEAT_JSCVT implements the functionality identified by 0b0001.

In Armv8.0, Armv8.1, and Armv8.2, the only permitted value is 0b0000.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is 0b0001.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is 0b0000.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
UNKNOWN																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Reserved, UNKNOWN.

Accessing ID_ISAR6_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_ISAR6_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b111

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && (IsFeatureImplemented(FEAT_FGT) || !IsZero(ID_ISAR6_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_ISAR6_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = ID_ISAR6_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = ID_ISAR6_EL1;
    elseif PSTATE.EL == EL3 then
        X[t, 64] = ID_ISAR6_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ID_PFR0_EL1 characteristics are:

Purpose

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with ID_PFR1_EL1.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch64 System register ID_PFR0_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ID_PFR0\[31:0\]](#).

Attributes

ID_PFR0_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																
RAS				DIT				AMU				CSV2				State3				State2				State1				State0				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:32]

Reserved, RES0.

RAS, bits [31:28]RAS Extension version. ~~Defined values are:~~

RAS	Meaning
0b0000	No RAS Extension.
0b0001	RAS Extension implemented.
0b0010	FEAT_RASv1p1 implemented. As 0b0001, and adds support for additional ERXMISC<m> System registers. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS and support for the optional RAS Timestamp Extension.
0b0011	FEAT_RASv2 implemented. As 0b0010, and requires that error records accessed through System registers conform to RAS System Architecture v2.

All other values are reserved.

FEAT_RAS implements the functionality identified by the value 0b0001.

FEAT_RASv1p1 implements the functionality identified by the value 0b0010.

FEAT_RASv2 implements the functionality identified by the value 0b0011.

In Armv8.0 and Armv8.1, the permitted values are 0b0000 and 0b0001.

From Armv8.4, when FEAT_DoubleFault is not implemented, and ERRIDR_EL1.NUM is 0, the permitted values are IMPLEMENTATION DEFINED 0b0001 or 0b0010.

From Armv8.2, the only permitted value is 0b00000b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented or ERRIDR_EL1.NUM is nonzero, the value 0b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented, the only permitted value is 0b0010.

Note

When the value of this field is 0b0001, ID_PFR2_EL1.RAS_frac indicates whether FEAT_RASv1p1 is implemented.

DIT, bits [27:24]

Data Independent Timing. Defined values are:

DIT	Meaning
0b0000	AArch32 does not guarantee constant execution time of any instructions.
0b0001	AArch32 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.

All other values are reserved.

FEAT_DIT implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

AMU, bits [23:20]

Indicates support for Activity Monitors Extension. Defined values are:

AMU	Meaning
0b0000	Activity Monitors Extension is not implemented.
0b0001	FEAT_AMUv1 is implemented.
0b0010	FEAT_AMUv1p1 is implemented. As 0b0001 and adds support for virtualization of the activity monitor event counters.

All other values are reserved.

FEAT_AMUv1 implements the functionality identified by the value 0b0001.

FEAT_AMUv1p1 implements the functionality identified by the value 0b0010.

In Armv8.0, the only permitted value is 0b0000.

In Armv8.4, the permitted values are 0b0000 and 0b0001.

From Armv8.6, the permitted values are 0b0000, 0b0001, and 0b0010.

CSV2, bits [19:16]

Speculative use of out of context branch targets. Defined values are:

CSV2	Meaning
0b0000	The implementation does not disclose whether FEAT_CSV2 is implemented.
0b0001	FEAT_CSV2 is implemented, but FEAT_CSV2_1p1 is not implemented.
0b0010	FEAT_CSV2_1p1 is implemented.

All other values are reserved.

FEAT_CSV2 implements the functionality identified by the value 0b0001.

FEAT_CSV2_1p1 implements the functionality identified by the value 0b0010.

From Armv8.5, the permitted values are 0b0001 and 0b0010.

State3, bits [15:12]

T32EE instruction set support. Defined values are:

State3	Meaning
0b0000	Not implemented.
0b0001	T32EE instruction set implemented.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

State2, bits [11:8]

Jazelle extension support. Defined values are:

State2	Meaning
0b0000	Not implemented.
0b0001	Jazelle extension implemented, without clearing of JOSCR.CV on exception entry.
0b0010	Jazelle extension implemented, with clearing of JOSCR.CV on exception entry.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0001.

State1, bits [7:4]

T32 instruction set support. Defined values are:

State1	Meaning
0b0000	T32 instruction set not implemented.
0b0001	T32 encodings before the introduction of Thumb-2 technology implemented: <ul style="list-style-type: none"> • All instructions are 16-bit. • A BL or BLX is a pair of 16-bit instructions. • 32-bit instructions other than BL and BLX cannot be encoded.
0b0011	T32 encodings after the introduction of Thumb-2 technology implemented, for all 16-bit and 32-bit T32 basic instructions.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0011.

State0, bits [3:0]

A32 instruction set support. Defined values are:

State0	Meaning
0b0000	A32 instruction set not implemented.
0b0001	A32 instruction set implemented.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0001.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
																	UNKNOWN																
																	UNKNOWN																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:0]

Reserved, UNKNOWN.

Accessing ID_PFR0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ID_PFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b00001	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_PFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_PFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_PFR0_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ISR_EL1, Interrupt Status Register

The ISR_EL1 characteristics are:

Purpose

Shows the pending status of the IRQ, FIQ, or SError interrupt.

When executing at EL2, EL3 or Secure EL1 when [SCR_EL3.EEL2](#) == 0b0, this shows the pending status of the physical IRQ, FIQ, or SError interrupts.

When executing at either Non-secure EL1 or at Secure EL1 when [SCR_EL3.EEL2](#) == 0b1:

- If the [HCR_EL2](#).{IMO,FMO,AMO} bit has a value of 1, the corresponding ISR_EL1.{I,F,A} bit shows the pending status of the virtual IRQ, FIQ, or SError.
- If the [HCR_EL2](#).{IMO,FMO,AMO} bit has a value of 0, the corresponding ISR_EL1.{I,F,A} bit shows the pending status of the physical IRQ, FIQ, or SError.

Configuration

AArch64 System register ISR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ISR\[31:0\]](#).

Attributes

ISR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0										IS		FS		A		I		F		RES0											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:11]

Reserved, RES0.

IS, bit [10]

When FEAT_NMI is implemented:

IRQ with Superpriority pending bit. Indicates whether an IRQ interrupt with Superpriority is pending.

IS	Meaning
0b0	No pending IRQ with Superpriority.
0b1	An IRQ interrupt with Superpriority is pending.

Otherwise:

Reserved, RES0.

FS, bit [9]**When FEAT_NMI is implemented:**

FIQ with Superpriority pending bit. Indicates whether an FIQ interrupt with Superpriority is pending.

FS	Meaning
0b0	No pending FIQ with Superpriority.
0b1	An FIQ interrupt with Superpriority is pending.

Otherwise:

Reserved, RES0.

A, bit [8]

SError interrupt pending bit. Indicates whether an SError interrupt is pending.

A	Meaning
0b0	No pending SError.
0b1	An SError interrupt is pending.

If the SError interrupt is edge-triggered, this field is cleared to zero when the physical SError interrupt is taken.

I, bit [7]

IRQ pending bit. Indicates whether an IRQ interrupt is pending.

I	Meaning
0b0	No pending IRQ.
0b1	An IRQ interrupt is pending.

Note

This bit indicates the presence of a pending IRQ interrupt regardless of whether the interrupt has Superpriority.

F, bit [6]

FIQ pending bit. Indicates whether an FIQ interrupt is pending.

F	Meaning
0b0	No pending FIQ.
0b1	An FIQ interrupt is pending.

Note

This bit indicates the presence of a pending FIQ interrupt regardless of whether the interrupt has Superpriority.

Bits [5:0]

Reserved, RES0.

Accessing ISR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ISR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.ISR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ISR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ISR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ISR_EL1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

LORC_EL1, LORegion Control (EL1)

The LORC_EL1 characteristics are:

Purpose

Enables and disables LORegions, and selects the current LORegion descriptor.

Configuration

This register is present only when FEAT_LOR is implemented. Otherwise, direct accesses to LORC_EL1 are UNDEFINED.

If no LORegion descriptors are supported by the PE, then this register is RES0.

Attributes

LORC_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																																
RES0																					DS										RES0	EN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:10]

Reserved, RES0.

DS, bits [9:2]

Descriptor Select. Selects the current LORegion descriptor accessed by [LORSA_EL1](#), [LOREA_EL1](#), and [LORN_EL1](#).

The number of LORegion descriptors in IMPLEMENTATION DEFINED. The maximum number of LORegion descriptors supported is 256. If the number is less than 256, then bits[63:M+2] are RES0, where M is $\text{Log}_2(\text{Number of LORegion descriptors supported by the implementation})$.

If this field points to an LORegion descriptor that is not supported by an implementation, then the registers [LORN_EL1](#), [LOREA_EL1](#), and [LORSA_EL1](#) are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [1]

Reserved, RES0.

EN, bit [0]

Enable. Indicates whether LORegions are enabled.

EN	Meaning
0b0	Disabled. Memory accesses do not match any LORegions.
0b1	Enabled. Memory accesses may match a LORegion.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing LORC_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, LORC_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.LORC_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORC_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORC_EL1;
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            X[t, 64] = LORC_EL1;

```

MSR LORC_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.LORC_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            LORC_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                LORC_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LORC_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

LOREA_EL1, LORegion End Address (EL1)

The LOREA_EL1 characteristics are:

Purpose

Holds the physical address of the end of the LORegion described in the current LORegion descriptor selected by [LORC_EL1](#).DS.

Configuration

This register is present only when FEAT_LOR is implemented. Otherwise, direct accesses to LOREA_EL1 are UNDEFINED.

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC_EL1](#).DS points to a LORegion that is not supported by the PE.

Attributes

LOREA_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								EA[55:52]EA[51:48]EA[51:48]EA[47:16]								EA[47:16]															
EA[47:16]																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Any of the fields in this register are permitted to be cached in a TLB.

Bits [63:5652]

Reserved, RES0.

EA[55:52], bits [55:52] When FEAT_D128 is implemented:

Extension to EA[47:16]. For more information, see EA[47:16].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA[51:48], bits [51:48] When FEAT_LPA is implemented:

Extension to EA[47:16]. For more information, see EA[47:16].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EA[47:16], bits [47:16]

Bits [47:16] of the end physical address of an LORegion described in the current LORegion descriptor selected by [LORC_EL1](#).DS. Bits[15:0] of this address are 0xFFFF. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

When FEAT_LPA is implemented and 52-bit addresses are in use, EA[51:48] form bits [51:48] of the end physical address of the LORegion. Otherwise, when 52-bit addresses are not in use, EA[51:48] is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [15:0]

Reserved, RES0.

Accessing LOREA_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, LOREA_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.LOREA_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LOREA_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LOREA_EL1;
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            X[t, 64] = LOREA_EL1;

```

MSR LOREA_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.LOREA_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            LOREA_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                LOREA_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LOREA_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

LORID_EL1, LORegionID (EL1)

The LORID_EL1 characteristics are:

Purpose

Indicates the number of LORegions and LORegion descriptors supported by the PE.

Configuration

This register is present only when FEAT_LOR is implemented. Otherwise, direct accesses to LORID_EL1 are UNDEFINED.

If no LORegion descriptors are implemented, then the registers [LORC_EL1](#), [LORN_EL1](#), [LOREA_EL1](#), and [LORSA_EL1](#) are RES0.

Attributes

LORID_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0								LD								RES0								LR							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:24]

Reserved, RES0.

LD, bits [23:16]

Number of LORegion descriptors supported by the PE. This is an 8-bit binary number.

Bits [15:8]

Reserved, RES0.

LR, bits [7:0]

Number of LORegions supported by the PE. This is an 8-bit binary number.

Note

If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease.

Accessing LORID_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, LORID_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = LORID_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

LORN_EL1, LORegion Number (EL1)

The LORN_EL1 characteristics are:

Purpose

Holds the number of the LORegion described in the current LORegion descriptor selected by [LORC_EL1](#).DS.

Configuration

This register is present only when FEAT_LOR is implemented. Otherwise, direct accesses to LORN_EL1 are UNDEFINED.

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC_EL1](#).DS points to a LORegion that is not supported by the PE.

Attributes

LORN_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																
RES0																Num																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Any of the fields in this register are permitted to be cached in a TLB.

Bits [63:8]

Reserved, RES0.

Num, bits [7:0]

Number of the LORegion described in the current LORegion descriptor selected by [LORC_EL1](#).DS.

The maximum number of LORegions supported by the PE is 256. If the maximum number is less than 256, then bits[8:N] are RES0, where N is (Log₂(Number of LORegions supported by the PE)).

If this field points to a LORegion that is not supported by the PE, then the current LORegion descriptor does not match any LORegion.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing LORN_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, LORN_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGRTR_EL2.LORN_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORN_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORN_EL1;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        UNDEFINED;
    else
        X[t, 64] = LORN_EL1;

```

MSR LORN_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.LORN_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            LORN_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                LORN_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LORN_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

LORSA_EL1, LORegion Start Address (EL1)

The LORSA_EL1 characteristics are:

Purpose

Indicates whether the current LORegion descriptor selected by [LORC_EL1](#).DS is enabled, and holds the physical address of the start of the LORegion.

Configuration

This register is present only when FEAT_LOR is implemented. Otherwise, direct accesses to LORSA_EL1 are UNDEFINED.

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC_EL1](#).DS points to a LORegion that is not supported by the PE.

Attributes

LORSA_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								SA																							
SA																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																Valid															

Any of the fields in this register are permitted to be cached in a TLB.

Bits [63:**56****52**]

Reserved, RES0.

SA, bits [**55****51**:16]

SA encoding when **FEAT_D128****FEAT_LPA** is implemented

39	38	37	36	35	34	33	32																												
SA																																			
SA																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																SA																			

SA, bits [**39****35**:0]

Bits [**55****51**:16] of the start physical address of the LORegion described in the current LORegion descriptor selected by [LORC_EL1](#).DS.

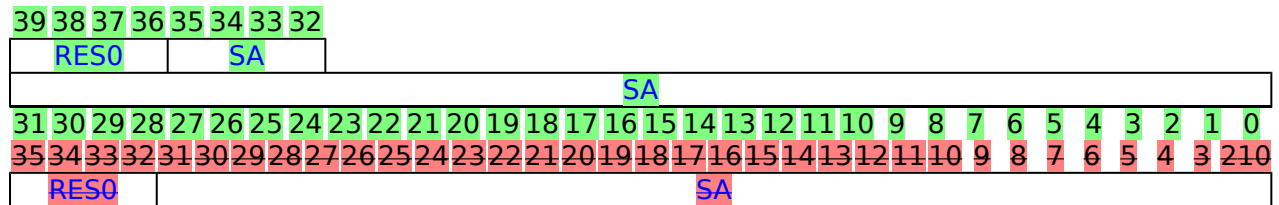
Bits[15:0] of this address are 0x0000.

For implementations with fewer than 5652 physical address bits, the corresponding upper bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SA encoding when FEAT_LPA is implemented and FEAT_D128 is not implemented



Bits [3935:3632]

Reserved, RES0.

SA, bits [3531:0]

Bits [5147:16] of the start physical address of the LORegion described in the current LORegion descriptor selected by LORC_EL1.DS.

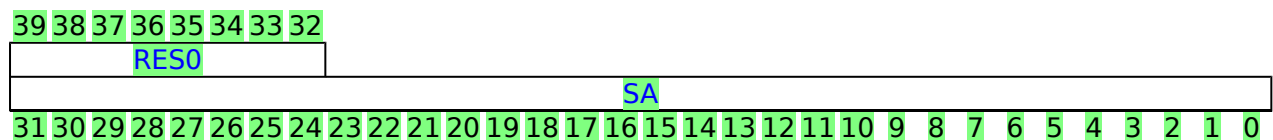
Bits[15:0] of this address are 0x0000.

For implementations with fewer than 5248 physical address bits, the corresponding upper bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SA encoding when FEAT_LPA is not implemented



Bits [39:32]

Reserved, RES0.

SA, bits [31:0]

Bits [47:16] of the start physical address of the LORegion described in the current LORegion descriptor selected by LORC_EL1.DS.

Bits[15:0] of this address are 0x0000.

For implementations with fewer than 48 physical address bits, the corresponding upper bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [15:1]

Reserved, RES0.

Valid, bit [0]

Indicates whether the current LORegion descriptor is enabled.

Valid	Meaning
0b0	LORegion descriptor is disabled.
0b1	LORegion descriptor is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing LORSA_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, LORSA_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.LORSA_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORSA_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = LORSA_EL1;
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            X[t, 64] = LORSA_EL1;

```

MSR LORSA_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elsif SCR_EL3.NS == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.LORSA_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            LORSA_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            LORSA_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LORSA_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MAIR2_EL1, Extended Memory Attribute Indirection Register (EL1)

The MAIR2_EL1 characteristics are:

Purpose

Provides the memory attribute encodings corresponding to the possible AttrIndx values in a VMSAv8-64 or VMSAv9-128 translation table entry for stage 1 translations at EL1.

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to MAIR2_EL1 are UNDEFINED.

Attributes

MAIR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Attr<n>, bits [8n+7:8n], for n = 7 to 0

Memory Attribute encoding.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 1, AttrIndx[2:0] gives the value of <n> in Attr<n>.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 0, see MAIR_ELx.Attr

Attr is encoded as follows:

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iii != 0000)	Normal memory. See encoding of 'oooo' and 'iii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where xxxx != 0000 and xxxx != 0100 and xxxx != 1010 and xxxx != 1111	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iii' is encoded as follows:

'iii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x280];
    else
        X[t, 64] = MAIR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR2_EL2;
    else
        X[t, 64] = MAIR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR2_EL1;

```

MSR MAIR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x280] = X[t, 64];
    else
        MAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        MAIR2_EL2 = X[t, 64];
    else
        MAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR2_EL1 = X[t, 64];

```

MRS <Xt>, MAIR2_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x280];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = MAIR2_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR2_EL1;
    else
        UNDEFINED;

```

MSR MAIR2_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x280] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            MAIR2_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        MAIR2_EL1 = X[t, 64];
    else
        UNDEFINED;

```

no old file

htmldiff from-

(new)

MAIR2_EL2, Extended Memory Attribute Indirection Register (EL2)

The MAIR2_EL2 characteristics are:

Purpose

Provides the memory attribute encodings corresponding to the possible AttrIndx values in a VMSAv8-64 or VMSAv9-128 translation table entry for stage 1 translations at EL1.

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to MAIR2_EL2 are UNDEFINED.

Attributes

MAIR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Attr<n>, bits [8n+7:8n], for n = 7 to 0

Memory Attribute encoding.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 1, AttrIndx[2:0] gives the value of <n> in Attr<n>.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 0, see MAIR_ELx.Attr

Attr is encoded as follows:

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iii != 0000)	Normal memory. See encoding of 'oooo' and 'iii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where xxxx != 0000 and xxxx != 0100 and xxxx != 1010 and xxxx != 1111	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iii' is encoded as follows:

'iii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR2_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name MAIR2_EL2 or MAIR2_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MAIR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR2_EL2;

```

MSR MAIR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MAIR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR2_EL2 = X[t, 64];

```

MRS <Xt>, MAIR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x280];
    else
        X[t, 64] = MAIR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR2_EL2;
    else
        X[t, 64] = MAIR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR2_EL1;

```

MSR MAIR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nMAIR2_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x280] = X[t, 64];
    else
        MAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.AIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.AIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        MAIR2_EL2 = X[t, 64];
    else
        MAIR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR2_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

MAIR2_EL3, Extended Memory Attribute Indirection Register (EL3)

The MAIR2_EL3 characteristics are:

Purpose

Provides the memory attribute encodings corresponding to the possible AttrIndx values in a VMSAv8-64 or VMSAv9-128 translation table entry for stage 1 translations at EL1.

Configuration

This register is present only when FEAT_AIE is implemented. Otherwise, direct accesses to MAIR2_EL3 are UNDEFINED.

Attributes

MAIR2_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Attr<n>, bits [8n+7:8n], for n = 7 to 0

Memory Attribute encoding.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 1, AttrIndx[2:0] gives the value of <n> in Attr<n>.

When stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a VMSAv8-64 or VMSAv9-128 translation table entry is 0, see MAIR_ELx.Attr

Attr is encoded as follows:

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iii != 0000)	Normal memory. See encoding of 'oooo' and 'iii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where xxxx != 0000 and xxxx != 0100 and xxxx != 1010 and xxxx != 1111	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iiii' is encoded as follows:

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iiii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR2_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR2_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0001	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR2_EL3;
```

MSR MAIR2_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0001	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MAIR2_EL3 = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding of 'oooo' and 'iiii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where (xxxx != 0000 and, xxxx != 0100 and, xxxx != 1010 and, xxxx != 1111)	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iiii' is encoded as follows:

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iiii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic MAIR_EL1 or MAIR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGRTR_EL2.MAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x140];
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR_EL2;
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR_EL1;

```

MSR MAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.MAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x140] = X[t, 64];
    else
        MAIR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            MAIR_EL2 = X[t, 64];
        else
            MAIR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        MAIR_EL1 = X[t, 64];

```

MRS <Xt>, MAIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x140];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            X[t, 64] = MAIR_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL3 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
            X[t, 64] = MAIR_EL1;
        else
            UNDEFINED;

```

MSR MAIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x140] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MAIR_EL2, Memory Attribute Indirection Register (EL2)

The MAIR_EL2 characteristics are:

Purpose

Provides the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations at EL2.

Configuration

AArch64 System register MAIR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HMAIR0\[31:0\]](#).

AArch64 System register MAIR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HMAIR1\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

MAIR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAIR_EL2 is permitted to be cached in a TLB.

Attr<n>, bits [8n+7:8n], for n = 7 to 0

Memory The Attribute memory encoding, attribute encoding for an AttrIndx[2:0] entry in a Long descriptor format translation table entry, where AttrIndx[2:0] gives the value of <n> in Attr<n>.

When FEAT_AIE is implemented and stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a Long descriptor format translation table entry is 0, or when FEAT_AIE is not implemented, AttrIndx[2:0] gives the value of <n> in Attr<n>.

When FEAT_AIE is implemented and stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a Long descriptor format translation table entry is 1, see MAIR2_ELx.Attr

Attr is encoded as follows:

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding of 'oooo' and 'iiii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where (xxxx != 0000 and, xxxx != 0100 and, xxxx != 1010 and, xxxx != 1111)	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iiii' is encoded as follows:

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iiii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic MAIR_EL2 or MAIR_EL1 is not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MAIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR_EL2;

```

MSR MAIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    MAIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR_EL2 = X[t, 64];

```

MRS <Xt>, MAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.MAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x140];
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR_EL2;
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR_EL1;

```

MSR MAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.MAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x140] = X[t, 64];
    else
        MAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        MAIR_EL2 = X[t, 64];
    else
        MAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5807: 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding of 'dd' for the type of Device memory. Otherwise, UNPREDICTABLE.
0b0000dd1x	UNPREDICTABLE.
0boooooiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding of 'oooo' and 'iiii' for the type of Normal Memory.
0b01000000	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b10100000	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise, UNPREDICTABLE.
0b11110000	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise, UNPREDICTABLE.
0bxxxx0000, where (xxxx != 0000 and, xxxx != 0100 and, xxxx != 1010 and, xxxx != 1111)	UNPREDICTABLE.

'dd' is encoded as follows:

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iiii' is encoded as follows:

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iiii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MAIR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MAIR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR_EL3;
```

MSR MAIR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MAIR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MDCR_EL2, Monitor Debug Configuration Register (EL2)

The MDCR_EL2 characteristics are:

Purpose

Provides EL2 configuration options for self-hosted debug and the Performance Monitors Extension.

Configuration

AArch64 System register MDCR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HDCR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

MDCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
RES0																				EBWE	HPMFZS	RES0	PMEE	
RES0	HPMFZ0	MTPME	TDCC	HLPE	E2TB	HCCD	RES0	TTRF	RES0	HPMD	RES0	TPMS	E2PB		TDRA		TDOSA	TDATA	TDE	HPM				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7

Bits [63:44]37

Reserved, RES0.

EBWE, bit [43]
When FEAT_Debugv8p9 is implemented:

Extended Breakpoint and Watchpoint Enable. Enables use of additional breakpoints or watchpoints.

EBWE	Meaning
0b0	The Effective value of MDSCR_EL1.EBWE is 0.
0b1	The Effective value of MDSCR_EL1.EBWE is not affected by this bit.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and [MDSELR_EL1](#) is implemented as RAZ/WI.

If EL2 is not implemented or EL2 is disabled in the current Security state, then the Effective value of this field is 1, other than for a direct read of the register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [42]

Reserved, RES0.

PMEE, bits [41:40]**When FEAT_EBEP is implemented:**

Performance Monitors Exception Enable. Controls the generation of **PMUIRQ** signal and PMU exception at EL0, EL1, and EL2.

PMEE	Meaning
0b00	PMUIRQ signal is enabled, and PMU exception is disabled.
0b01	PMUIRQ signal and PMU exception are both controlled by PMECR_EL1.PMEE .
0b10	PMUIRQ signal is disabled, and PMU exception is disabled.
0b11	PMUIRQ signal is disabled, and PMU exception is enabled.

If EL2 is not implemented or EL2 is disabled in the current Security state, then the Effective value of this field is 0b01, other than for a direct read of the register.

This field is ignored by the PE when all of the following are true:

- [EL3](#) is implemented.
- [MDCR_EL3.PMEE](#) != 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [39:37]

Reserved, RES0.

HPMFZS, bit [36]**When FEAT_SPEv1p2 is implemented:**

Hyp Performance Monitors Freeze-on-SPE event. Stop counters when [PMBLIMITR_EL1](#).{PMFZ, E} == {1, 1} and [PMBSR_EL1](#).S == 1.

HPMFZS	Meaning
0b0	Do not freeze on Statistical Profiling Buffer Management event.
0b1	Event counters do not count following a Statistical Profiling Buffer Management event.

If MDCR_EL2.HPMN is less than [PMCR_ELO](#).N, this field affects the operation of event counters in the range [MDCR_EL2.HPMN .. ([PMCR_ELO](#).N-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [35:30]

Reserved, RES0.

HPMFZO, bit [29]

When FEAT_PMUv3p7 is implemented:

Hyp Performance Monitors Freeze-on-overflow. Stop event counters on overflow.

HPMFZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counters do not count when PMOVSLR_EL0 [(PMCR_EL0 .N-1):MDCR_EL2.HPMN] is nonzero.

If MDCR_EL2.HPMN is less than [PMCR_EL0](#).N, this field affects the operation of event counters in the range [MDCR_EL2.HPMN .. ([PMCR_EL0](#).N-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MTPME, bit [28]

When FEAT_MTPMU is implemented and EL3 is not implemented:

Multi-threaded PMU Enable. Enables use of the [PMEVTYPER<n>_EL0](#).MT bits.

MTPME	Meaning
0b0	FEAT_MTPMU is disabled. The Effective value of PMEVTYPER<n>_EL0 .MT is zero.
0b1	PMEVTYPER<n>_EL0 .MT bits not affected by this field.

If FEAT_MTPMU is disabled for any other PE in the system that has the same level 1 Affinity as the PE, it is IMPLEMENTATION DEFINED whether the PE behaves as if this field is 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 1.

Otherwise:

Reserved, RES0.

TDCC, bit [27]**When FEAT_FGT is implemented:**

Trap DCC. Traps use of the Debug Comms Channel at EL1 and EL0 to EL2.

TDCC	Meaning
0b0	This control does not cause any register accesses to be trapped.
0b1	If EL2 is implemented and enabled in the current Security state, accesses to the DCC registers at EL1 and EL0 generate a Trap exception to EL2, unless the access also generates a higher priority exception. Traps on the DCC data transfer registers are ignored when the PE is in Debug state.

The DCC registers trapped by this control are:

AArch64: [OSDTRRX_EL1](#), [OSDTRTX_EL1](#), [MDCCSR_EL0](#), [MDCCINT_EL1](#), and, when the PE is in Non-debug state, [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#).

AArch32: [DBGDTRRXext](#), [DBGDTRTXext](#), [DBGDSCRint](#), [DBGDCCINT](#), and, when the PE is in Non-debug state, [DBGDTRRXint](#) and [DBGDTRTXint](#).

The traps are reported with EC syndrome value:

- 0x05 for trapped AArch32 MRC and MCR accesses with coproc == 0b1110.
- 0x06 for trapped AArch32 LDC to [DBGDTRTXint](#) and STC from [DBGDTRRXint](#).
- 0x18 for trapped AArch64 MRS and MSR accesses.

When the PE is in Debug state, MDCR_EL2.TDCC does not trap any accesses to:

AArch64: [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#).

AArch32: [DBGDTRRXint](#) and [DBGDTRTXint](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HLP, bit [26]**When FEAT_PMUv3p5 is implemented:**

Hypervisor Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit.

HLP	Meaning
0b0	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0 [31:0].
0b1	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0 [63:0].

If MDCR_EL2.HPMN is less than [PMCR_EL0](#).N, this bit affects the operation of event counters in the range [MDCR_EL2.HPMN..([PMCR_EL0](#).N-1)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E2TB, bits [25:24]**When FEAT_TRBE is implemented:**

EL2 Trace Buffer.

If EL2 is implemented and enabled in the Trace Buffer owning Security state, controls the owning translation regime.

If EL2 is implemented and enabled in the current Security state, controls access to Trace Buffer control registers from EL1.

E2TB	Meaning
0b00	If EL2 is implemented and enabled in the Trace Buffer owning Security state, the Trace Buffer owning Exception level is EL2. Otherwise, the Trace Buffer owning Exception level is EL1 and, if <code>TraceBufferEnabled() == TRUE</code> , tracing is prohibited at EL2. If EL2 is implemented and enabled in the current Security state, accesses to Trace Buffer control registers at EL1 generate a Trap exception to EL2.
0b10	Trace Buffer owning Exception level is EL1. If <code>TraceBufferEnabled() == TRUE</code> , tracing is prohibited at EL2. If EL2 is implemented and enabled in the current Security state, accesses to Trace Buffer control registers at EL1 generate a Trap exception to EL2.
0b11	Trace Buffer owning Exception level is EL1. If <code>TraceBufferEnabled() == TRUE</code> , tracing is prohibited at EL2.

All other values are reserved.

The Trace Buffer control registers trapped by this control are: [TRBBASER_EL1](#), [TRBLIMITR_EL1](#), [TRBMAR_EL1](#), [TRBPTR_EL1](#), [TRBSR_EL1](#), and [TRBTRG_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HCCD, bit [23]**When FEAT_PMUv3p5 is implemented:**

Hypervisor Cycle Counter Disable. Prohibits [PMCCNTR_EL0](#) from counting at EL2.

HCCD	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR_EL0 is prohibited at EL2.

This field does not affect the CPU_CYCLES event or any other event that counts cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [22:20]

Reserved, RES0.

TTRF, bit [19]**When FEAT_TRF is implemented:**

Traps use of the Trace Filter Control registers at EL1 to EL2, as follows:

- Access to [TRFCR_EL1](#) is trapped to EL2, reported using EC syndrome value 0x18.
- Access to [TRFCR](#) is trapped to EL2, reported using EC syndrome value 0x03.

TTRF	Meaning
0b0	Accesses to the specified registers at EL1 are not affected by this control.
0b1	Accesses to the specified registers at EL1 generate a trap exception to EL2 when EL2 is enabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [18]

Reserved, RES0.

HPMD, bit [17]**When FEAT_PMUv3p1 is implemented and FEAT_Debugv8p2 is implemented:**

Guest Performance Monitors Disable. Controls event counting by some event counters at EL2.

HPMD	Meaning
0b0	Event counting and PMCCNTR_EL0 are not affected by this mechanism.
0b1	Event counting by some event counters is prohibited at EL2. If PMCR_EL0 .DP is 1, PMCCNTR_EL0 is disabled at EL2. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.

If MDCR_EL2.HPMN is not 0, this field affects the operation of event counters in the range [0 .. (MDCR_EL2.HPMN-1)].

This field does not affect the operation of other event counters.

If [PMCR_EL0](#).DP is 1, this field affects [PMCCNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_PMUv3p1 is implemented:

Guest Performance Monitors Disable. Controls event counting by some event counters at EL2.

HPMD	Meaning
0b0	Event counting and PMCCNTR_ELO are not affected by this mechanism.
0b1	If <code>ExternalSecureNoninvasiveDebugEnabled()</code> is FALSE, event counting by some event counters is prohibited at EL2, and if PMCR_ELO .DP is 1, PMCCNTR_ELO is disabled at EL2.

If `ExternalSecureNoninvasiveDebugEnabled()` is TRUE, this field does not affect the event counters and does not affect [PMCCNTR_ELO](#).

Otherwise:

- If MDCR_EL2.HPMN is not 0, this field affects the operation of event counters in the range [0 .. (MDCR_EL2.HPMN-1)].
- This field does not affect the operation of other event counters.
- If [PMCR_ELO](#).DP is 1, this field affects [PMCCNTR_ELO](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [16:15]

Reserved, RES0.

TPMS, bit [14]**When FEAT_SPE is implemented:**

Trap Performance Monitor Sampling. If EL2 is implemented and enabled in the current Security state, controls access to Statistical Profiling control registers from EL1.

TPMS	Meaning
0b0	Do not trap Statistical Profiling controls to EL2.
0b1	If EL2 is implemented and enabled in the current Security state, accesses to Statistical Profiling control registers at EL1 generate a Trap exception to EL2.

The Statistical Profiling control registers trapped by this control are:

- [PMSCR_EL1](#), [PMSEVFR_EL1](#), [PMSFCR_EL1](#), [PMSICR_EL1](#), [PMSIDR_EL1](#), [PMSIRR_EL1](#), and [PMSLATER_EL1](#).
- If FEAT_SPEv1p2 is implemented, [PMSNEVFR_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E2PB, bits [13:12]**When FEAT_SPE is implemented:**

EL2 Profiling Buffer. If EL2 is implemented and enabled in the Profiling Buffer owning Security state, this field controls the owning translation regime. If EL2 is implemented and enabled in the current Security state, this field controls access to Profiling Buffer control registers from EL1.

E2PB	Meaning
0b00	If EL2 is implemented and enabled in the Profiling Buffer owning Security state, the Profiling Buffer uses the EL2 or EL2&0 stage 1 translation regime. Otherwise the Profiling Buffer uses the EL1&0 stage 1 translation regime. If EL2 is implemented and enabled in the current Security state, accesses to Profiling Buffer control registers at EL1 generate a Trap exception to EL2.
0b10	Profiling Buffer uses the EL1&0 stage 1 translation regime. If EL2 is implemented and enabled in the current Security state, accesses to Profiling Buffer control registers at EL1 generate a Trap exception to EL2.
0b11	Profiling Buffer uses the EL1&0 stage 1 translation regime. Accesses to Profiling Buffer control registers at EL1 are not trapped to EL2.

All other values are reserved.

The Profiling Buffer control registers trapped by this control are: [PMBLIMITR_EL1](#), [PMBPTR_EL1](#), and [PMBSR_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TDRA, bit [11]

Trap Debug ROM Address register access. Traps System register accesses to the Debug ROM registers to EL2 when EL2 is enabled in the current Security state as follows:

- If EL1 is using AArch64 state, accesses to [MDRAR_EL1](#) are trapped to EL2, reported using EC syndrome value 0x18.
- If EL0 or EL1 is using AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x05 and MRRC or MCRR accesses are trapped to EL2, reported using EC syndrome value 0x0C:
 - [DBGDRAR](#), [DBGDSAR](#).

TDRA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 and EL1 System register accesses to the Debug ROM registers are trapped to EL2 when EL2 is enabled in the current Security state, unless it is trapped by the following: <ul style="list-style-type: none"> DBGDSCRext.UDCCdis. MDSCR_EL1.TDCC.

This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:

- [MDCR_EL2](#).TDE == 1.
- [HCR_EL2](#).TGE == 1.

Note

EL2 does not provide traps on debug register accesses through the optional memory-mapped external debug interfaces.

System register accesses to the debug registers might have side-effects. When a System register access is trapped to EL2, no side-effects occur before the exception is taken to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDOSA, bit [10]

When FEAT_DoubleLock is implemented:

Trap debug OS-related register access. Traps EL1 System register accesses to the powerdown debug registers to EL2, from both Execution states as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18:
 - [OSLAR_EL1](#), [OSLSR_EL1](#), [OSDLR_EL1](#), and [DBGPRCR_EL1](#).
 - Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.
- In AArch32 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x05:
 - [DBGOSLSR](#), [DBGOSLAR](#), [DBGOSDLR](#), and [DBGPRCR](#).
 - Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 System register accesses to the powerdown debug registers are trapped to EL2 when EL2 is enabled in the current Security state.

Note

These registers are not accessible at EL0.

This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:

- [MDCR_EL2](#).TDE == 1.
- [HCR_EL2](#).TGE == 1.

System register accesses to the debug registers might have side-effects. When a System register access is trapped to EL2, no side-effects occur before the exception is taken to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Trap debug OS-related register access. Traps EL1 System register accesses to the powerdown debug registers to EL2, from both Execution states as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18:
 - [OSLAR_EL1](#), [OSLSR_EL1](#), and [DBGPRCR_EL1](#).
 - Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.
- In AArch32 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x05:
 - [DBGOSLSR](#), [DBGOSLAR](#), and [DBGPRCR](#).
 - Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

It is IMPLEMENTATION DEFINED whether accesses to [OSDLR_EL1](#) are trapped.

It is IMPLEMENTATION DEFINED whether accesses to [DBGOSDLR](#) are trapped.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 System register accesses to the powerdown debug registers are trapped to EL2 when EL2 is enabled in the current Security state.

Note

These registers are not accessible at EL0.

This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:

- [MDCR_EL2.TDE](#) == 1.
- [HCR_EL2.TGE](#) == 1.

Note

EL2 does not provide traps on debug register accesses through the optional memory-mapped external debug interfaces.

System register accesses to the debug registers might have side-effects. When a System register access is trapped to EL2, no side-effects occur before the exception is taken to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDA, bit [9]

Trap Debug Access. Traps EL0 and EL1 System register accesses to debug System registers that are not trapped by MDCR_EL2.TDRA or MDCR_EL2.TDOSA, as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2 reported using EC syndrome value 0x18:
 - [MDCCSR_EL0](#), [MDCCINT_EL1](#), [OSDTRRX_EL1](#), [MDSCR_EL1](#), [OSDTRTX_EL1](#), [OSECCR_EL1](#), [DBGBVR<n>_EL1](#), [DBGBCR<n>_EL1](#), [DBGWVR<n>_EL1](#), [DBGWCR<n>_EL1](#), [DBGCLAIMSET_EL1](#), [DBGCLAIMCLR_EL1](#), [DBGAUTHSTATUS_EL1](#).
 - When not in Debug state, [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), [DBGDTRTX_EL0](#).
- In AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x05.
 - [DBGDIDR](#), [DBGDSCRint](#), [DBGDCCINT](#), [DBGWFAR](#), [DBGVCR](#), [DBGDSCRext](#), [DBGDTRTXext](#), [DBGDTRRXext](#), [DBGBVR<n>](#), [DBGBCR<n>](#), [DBGBXVR<n>](#), [DBGWCR<n>](#), [DBGWVR<n>](#), [DBGCLAIMSET](#), [DBGCLAIMCLR](#), [DBGAUTHSTATUS](#), [DBGDEVID](#), [DBGDEVID1](#), [DBGDEVID2](#), [DBGOSECCR](#).
 - When not in Debug state, [DBGDTRRXint](#) and [DBGDTRTXint](#).
- In AArch32 state, STC accesses to [DBGDTRRXint](#) and LDC accesses to [DBGDTRTXint](#) are trapped to EL2, reported using EC syndrome value 0x06.

TDA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 or EL1 System register accesses to the debug registers are trapped from both Execution states to EL2 when EL2 is enabled in the current Security state, unless the access generates a higher priority exception.

Traps of AArch32 accesses to [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

Traps of AArch64 accesses to [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#) are ignored in Debug state.

This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:

- [MDCR_EL2.TDE](#) == 1
- [HCR_EL2.TGE](#) == 1

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDE, bit [8]

Trap Debug Exceptions. Controls routing of Debug exceptions, and defines the debug target Exception level, EL_D.

TDE	Meaning
0b0	The debug target Exception level is EL1.
0b1	If EL2 is enabled for the current Effective value of SCR_EL3.NS , the debug target Exception level is EL2, otherwise the debug target Exception level is EL1. The MDCR_EL2.{TDRA, TDOSA, TDA} fields are treated as being 1 for all purposes other than returning the result of a direct read of the register.

For more information, see 'Routing debug exceptions'.

This field is treated as being 1 for all purposes other than a direct read when [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HPME, bit [7]

When FEAT_PMuV3 is implemented:

[MDCR_EL2.HPMN..(N-1)] event counters enable.

HPME	Meaning
0b0	Event counters in the range [MDCR_EL2.HPMN.. PMCR_EL0.N-1] are disabled.
0b1	Event counters in the range [MDCR_EL2.HPMN.. PMCR_EL0.N-1] are enabled by PMCNTENSET_EL0 .

If MDCR_EL2.HPMN is less than [PMCR_EL0.N](#), this field affects the operation of event counters in the range [MDCR_EL2.HPMN..[PMCR_EL0.N-1](#)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TPM, bit [6]

When FEAT_PMuV3 is implemented:

Trap Performance Monitors accesses. Traps EL0 and EL1 accesses to all Performance Monitor registers to EL2 when EL2 is enabled in the current Security state, from both Execution states, as follows:

- In AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18:
 - [PMCR_EL0](#), [PMCNTENSET_EL0](#), [PMCNTENCLR_EL0](#), [PMOVSCLR_EL0](#), [PMSWINC_EL0](#), [PMSELR_EL0](#), [PMCEID0_EL0](#), [PMCEID1_EL0](#), [PMCCNTR_EL0](#), [PMXEVTYPER_EL0](#), [PMXVCNTR_EL0](#), [PMUSERENR_EL0](#), [PMINTENSET_EL1](#), [PMINTENCLR_EL1](#), [PMOVSSET_EL0](#), [PMEVCNTR<n>_EL0](#), [PMEVTYPER<n>_EL0](#), [PMCCFILTR_EL0](#).

- If FEAT_PMUv3p4 is implemented, [PMMIR_EL1](#)
- In AArch32 state, MRC or MCR accesses to the following registers are trapped to EL2 and reported using EC syndrome value 0x03, MRRC or MCRR accesses are trapped to EL2 and reported using EC syndrome value 0x04:
 - [PMCR](#), [PMCNTENSET](#), [PMCNTENCLR](#), [PMOVSr](#), [PMSWINC](#), [PMSELR](#), [PMCEID0](#), [PMCEID1](#), [PMCCNTR](#), [PMXEVTYPER](#), [PMXVCNTR](#), [PMUSERENR](#), [PMINTENSET](#), [PMINTENCLR](#), [PMOVSSET](#), [PMEVCNTR<n>](#), [PMEVTYPER<n>](#), [PMCCFILTR](#).
 - If FEAT_PMUv3p1 is implemented, [PMCEID2](#), and [PMCEID3](#).
 - If FEAT_PMUv3p4 is implemented, [PMMIR](#).

TPM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 and EL1 accesses to all Performance Monitor registers are trapped to EL2 when EL2 is enabled in the current Security state.

Note

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TPMCR, bit [5]**When FEAT_PMUv3 is implemented:**

Trap [PMCR_EL0](#) or [PMCR](#) accesses. Traps EL0 and EL1 accesses to EL2, when EL2 is enabled in the current Security state, as follows:

- In AArch64 state, accesses to [PMCR_EL0](#) are trapped to EL2, reported using EC syndrome value 0x18.
- In AArch32 state, accesses to [PMCR](#) are trapped to EL2, reported using EC syndrome value 0x03.

TPMCR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 and EL1 accesses to the specified registers are trapped to EL2 when EL2 is enabled in the current Security state, unless it is trapped by the following: <ul style="list-style-type: none"> • PMUSERENR.EN. • PMUSERENR_EL0.EN.

Note

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPMN, bits [4:0]**When FEAT_PMUv3 is implemented:**

Defines the number of event counters [PMEVCNTR<n>_EL0](#) and, if FEAT_PMUv3_SS is implemented, snapshot registers [PMEVCNTSVR<n>_EL1](#) that are accessible from EL3, EL2, EL1, and from EL0 if permitted.

Defines the number of event counters ~~that are accessible from EL3, EL2, EL1, and from EL0 if permitted.~~ [PMEVCNTR<n>_EL0](#) that are accessible from EL3, EL2, EL1, and from EL0 if permitted.

If HPMN is not 0 and is less than [PMCR_EL0.N](#), HPMN divides the event counters into a first range [0..(HPMN-1)], and a second range [HPMN..([PMCR_EL0.N](#)-1)].

If FEAT_HPMN0 is implemented and this field is 0, all event counters are in the second range and none are in the first range.

If HPMN is equal to [PMCR_EL0.N](#), all event counters are in the first range and none are in the second range.

For an event counter <n> in the first range:

- The counter is accessible from EL1, EL2, and EL3.
- The counter is accessible from EL0 if permitted by [PMUSERENR_EL0](#) or [PMUSERENR](#).
- If FEAT_PMUv3p5 is implemented, [PMCR_EL0.LP](#) or [PMCR.LP](#) determines whether the counter overflow flag is set on unsigned overflow of [PMEVCNTR<n>_EL0\[31:0\]](#) or [PMEVCNTR<n>_EL0\[63:0\]](#).
- [PMCR_EL0.E](#) and [PMCNTENSET_EL0\[n\]](#) enable the operation of event counter n.

For an event counter <n> in the second range:

- The counter is accessible from EL2 and EL3.
- If EL2 is disabled in the current Security state, the event counter is also accessible from EL1, and from EL0 if permitted by [PMUSERENR_EL0](#).
- If FEAT_PMUv3p5 is implemented, [MDCR_EL2.HLP](#) determines whether the counter overflow flag is set on unsigned overflow of [PMEVCNTR<n>_EL0\[31:0\]](#) or [PMEVCNTR<n>_EL0\[63:0\]](#).
- [MDCR_EL2.HPME](#) and [PMCNTENSET_EL0\[n\]](#) enable the operation of event counter n.

If HPMN is larger than [PMCR_EL0.N](#), or if FEAT_HPMN0 is not implemented and HPMN is 0, the following CONSTRAINED UNPREDICTABLE behaviors apply:

- The value returned by a direct read of [MDCR_EL2.HPMN](#) is UNKNOWN.
- One of the following behaviors:
 - An UNKNOWN number of counters are reserved for EL2 and EL3 use. That is, the PE behaves as if [MDCR_EL2.HPMN](#) is set to an UNKNOWN non-zero value less than or equal to [PMCR_EL0.N](#).
 - All counters are reserved for EL2 and EL3 use, meaning no counters are accessible from EL1 and EL0.

When FEAT_PMUv3_SS is implemented:

- For a snapshot register in the first range, the register is accessible from EL1, EL2, and EL3.
- ~~The value returned by a direct read of [MDCR_EL2.HPMN](#) is UNKNOWN.~~
- For a snapshot register in the second range, the register is accessible from EL2, and EL3. ~~Either:~~
 - ~~An UNKNOWN number of counters are reserved for EL2 and EL3 use. That is, the PE behaves as if [MDCR_EL2.HPMN](#) is set to an UNKNOWN non-zero value less than or equal to [PMCR_EL0.N](#).~~
 - ~~All counters are reserved for EL2 and EL3 use, meaning no counters are accessible from EL1 and EL0.~~
- If HPMN is larger than [PMCR_EL0.N](#), or if FEAT_HPMN0 is not implemented and HPMN is 0, then the above CONSTRAINED UNPREDICTABLE behaviors apply.

The reset behavior of this field is:

- On a Warm reset, this field resets to the value in [PMCR_EL0.N](#).

Otherwise:

Reserved, RES0.

Accessing MDCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MDCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDCR_EL2;

```

MSR MDCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
        MDCR_EL2 = X[t, 64];

```

(old)**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

MDCR_EL3, Monitor Debug Configuration Register (EL3)

The MDCR_EL3 characteristics are:

Purpose

Provides EL3 configuration options for self-hosted debug and the Performance Monitors Extension.

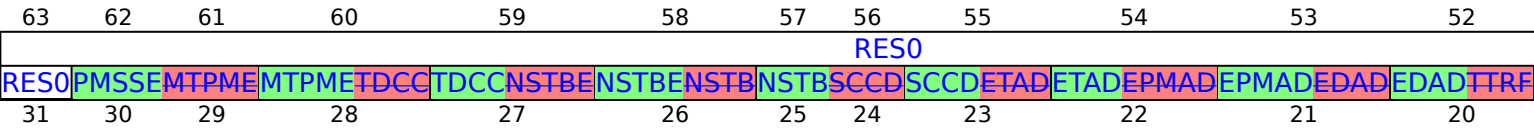
Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to MDCR_EL3 are UNDEFINED.

Attributes

MDCR_EL3 is a 64-bit register.

Field descriptions



Bits [63:5039]

Reserved, RES0.

ETBAD, bits [49:48]
When FEAT_TRBE_EXT is implemented:

External Trace Buffer Access Disable. Controls access to the Trace Buffer registers from an external debugger.

ETBAD	Meaning	Applies when
0b00	Non-secure accesses from an external debugger to Trace Buffer registers are prohibited. If FEAT_RME is implemented, Secure and Realm accesses from an external debugger to Trace Buffer registers are prohibited and Root accesses to Trace Buffer registers are allowed. If FEAT_RME is not implemented, Secure accesses to Trace Buffer registers are allowed.	
0b01	Secure and Non-secure accesses from an external debugger to Trace Buffer registers are prohibited. Root and Realm accesses to Trace Buffer registers are allowed.	When FEAT_RME is implemented
0b10	Realm and Non-secure accesses from an external debugger to Trace Buffer registers are prohibited. Root and Secure accesses to Trace Buffer registers are allowed.	When FEAT_RME is implemented
0b11	All accesses from an external debugger to Trace Buffer registers are allowed.	

If EL3 is not implemented, then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EnITE, bit [47]

When FEAT_ITE is implemented:

Enable access to Instrumentation trace registers. When disabled, accesses to Instrumentation trace registers generate a trap to EL3.

EnITE	Meaning
0b0	The following instructions at EL2 and EL1 are trapped to EL3, unless the instruction generates a higher priority exception: AArch64: MRS and MSR accesses to TRCITECR_EL1 , TRCITECR_EL2 , and TRCITECR_EL12 , reported with EC syndrome value 0x18.
0b1	Accesses of Instrumentation trace registers are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EPMSSAD, bits [46:45]**When FEAT_PMuV3_SS is implemented:**

External PMU Snapshot Access Disable. Controls access to the PMU Snapshot registers from an external debugger.

EPMSSAD	Meaning	Applies when
0b00	Non-secure accesses from an external debugger to PMU Snapshot registers are prohibited. If FEAT_RME is implemented, Secure and Realm accesses from an external debugger to PMU Snapshot registers are prohibited and Root accesses to PMU Snapshot registers are allowed. If FEAT_RME is not implemented, Secure accesses to PMU Snapshot registers are allowed.	
0b01	Secure and Non-secure accesses from an external debugger to PMU Snapshot registers are prohibited. Root and Realm accesses to PMU Snapshot registers are allowed.	When FEAT_RME is implemented
0b10	Realm and Non-secure accesses from an external debugger to PMU Snapshot registers are prohibited. Root and Secure accesses to PMU Snapshot registers are allowed.	When FEAT_RME is implemented
0b11	All accesses from an external debugger to PMU Snapshot registers are allowed.	

If EL3 is not implemented, then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EnPMSS, bit [44]**When FEAT_PMuV3_SS is implemented:**

Enable access to PMU Snapshot registers. When disabled, accesses to PMU Snapshot registers generate a trap to EL3.

EnPMSS	Meaning
0b0	Accesses of the specified PMU Snapshot registers at EL2 and EL1 are trapped to EL3, unless the instruction generates a higher priority exception.
0b1	Accesses of the specified PMU Snapshot registers are not trapped by this mechanism.

In AArch64 state, the instructions affected by this control are:

- MRS and MSR accesses to [PMCCNTSVR_EL1](#), [PMEVCNTSVR<n>_EL1](#), and [PMSSCR_EL1](#).
- If FEAT_PMuV3_ICNTR is implemented, MRS and MSR accesses to [PMICNTSVR_EL1](#).

Unless the instruction generates a higher priority exception, trapped instructions generate an exception to EL3.

Trapped instructions are reported using EC value 0x18.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EBWE, bit [43]**When FEAT_Debugv8p9 is implemented:**

Extended Breakpoint and Watchpoint Enable. Enables use of additional breakpoints or watchpoints, and enables a trap to EL3 on accesses to debug registers.

EBWE	Meaning
0b0	The Effective value of MDSCR_EL1.EBWE is 0. MRS and MSR accesses to MDSELR_EL1 at EL2 and EL1 are trapped to EL3, reported with EC syndrome value 0x18.
0b1	The Effective value of MDSCR_EL1.EBWE is not affected by this bit. Accesses to MDSELR_EL1 are not trapped by this mechanism.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and [MDSELR_EL1](#) is implemented as RAZ/WI.

If EL3 is not implemented, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnPMS3, bit [42]**When FEAT_SPEv1p4 is implemented or FEAT_SPE_FDS is implemented:**

Enable access to additional SPE registers. When disabled, accesses to additional SPE registers generate a trap to EL3.

EnPMS3	Meaning
0b0	The following instructions at EL2 and EL1 are trapped to EL3, unless the instruction generates a higher priority exception: AArch64: MRS and MSR accesses to PMSDSFR_EL1 , reported with EC syndrome value 0x18.
0b1	Accesses of additional SPE registers are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PMEE, bits [41:40]**When FEAT_EBEP is implemented:**

Performance Monitors Exception Enable. Controls the generation of **PMUIRQ** signal and PMU exception at all Exception levels.

PMEE	Meaning
0b00	PMUIRQ signal is enabled, and PMU exception is disabled.
0b01	PMUIRQ signal and PMU exception are both controlled by MDCR_EL2.PMEE .
0b10	PMUIRQ signal is disabled, and PMU exception is disabled.
0b11	PMUIRQ signal is disabled, and PMU exception is enabled.

If EL3 is not implemented, then the Effective value of this field is 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [39]

Reserved, RES0.

E3BREC, bit [38]**When FEAT_BRBEv1p1 is implemented:**

Branch Record Buffer EL3 Cold Reset Enable. With MDCR_EL3.E3BREW, controls branch recording at EL3.

E3BREC	Meaning
0b0	When MDCR_EL3.E3BREW == 0: Branch recording at EL3 is disabled. When MDCR_EL3.E3BREW == 1: Branch recording at EL3 is enabled.
0b1	When MDCR_EL3.E3BREW == 0: Branch recording at EL3 is enabled. When MDCR_EL3.E3BREW == 1: Branch recording at EL3 is disabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

E3BREW, bit [37]**When FEAT_BRBEv1p1 is implemented:**

Branch Record Buffer EL3 Warm Reset Enable. With MDCR_EL3.E3BREC, controls branch recording at EL3.

For a description of the values derived by evaluating MDCR_EL3.E3BREC and MDCR_EL3.E3BREW together, see MDCR_EL3.E3BREC.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EnPMSN, bit [36]**When FEAT_SPEv1p2 is implemented:**

Trap accesses to [PMSNEVFR_EL1](#). Controls access to Statistical Profiling [PMSNEVFR_EL1](#) System register from EL2 and EL1.

EnPMSN	Meaning
0b0	Accesses to PMSNEVFR_EL1 at EL2 and EL1 generate a Trap exception to EL3.
0b1	Do not trap PMSNEVFR_EL1 to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MPMX, bit [35]**When FEAT_PMUv3p7 is implemented:**

Monitor Performance Monitors Extended control. **WithIn conjunction with** MDCR_EL3.SPME, controls **PMUwhen operation event counters are enabled at EL3. EL3 and in other Secure Exception levels.**

MPMX	Meaning
0b0	CountersEvent are counting not affected by this mechanism and PMCCNTR_EL0 are not affected by this mechanism.
0b1	SomeEvent counting by some or all event counters areis prohibited from counting at EL3. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled at EL3. Otherwise, PMCCNTR_EL0 is not affected by this mechanism. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled at EL3. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.

TheIf countersEL2 affectedis byimplemented, thisMDCR_EL3.SPME field== are:1, and[MDCR_EL2](#).HPMN is less than [PMCR_EL0](#).N then all the following are true:

- If EL2 is implemented, MDCR_EL3.SPME is 1, and [MDCR_EL2](#).HPMN is not 0, this field affects the operation of event counters in the range [0 .. ([MDCR_EL2](#).HPMN-1)).] at EL3.**
- IfThis EL2field isdoes not implementedaffect orthe [MDCR_EL3.SPME](#)operation isof 0, allother event counters.**
- If FEAT_PMUv3_ICNTR is implemented, the instruction counter, [PMICNTR_EL0](#).**
- If [PMCR_EL0](#).DP is 1, this field affects the operation of [PMCCNTR_EL0](#) at EL3.**
- If [PMCR_EL0](#).DP is 1, the cycle counter, [PMCCNTR_EL0](#).**

If EL2 is not implemented, MDCR_EL3.SPME == 0, or [MDCR_EL2](#).HPMN is equal to [PMCR_EL0](#).N then this field affects the operation of all event counters at EL3, and if [PMCR_EL0](#).DP is 1, the operation of [PMCCNTR_EL0](#) at EL3.

OtherThe eventoperation countersof arethis notfield affectedapplies byeven thiswhen field.EL2 Whenis disabled in the current Security state. [PMCR_EL0](#).DP is 0, [PMCCNTR_EL0](#) is not affected by this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

MCCD, bit [34]**When FEAT_PMUv3p7 is implemented:**

Monitor Cycle Counter Disable. Prohibits the Cycle Counter, [PMCCNTR_EL0](#), from counting at EL3.

MCCD	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR_EL0 is prohibited at EL3.

This field does not affect the CPU_CYCLES event or any other event that counts cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

SBRBE, bits [33:32]**When FEAT_BRBE is implemented:**

Secure Branch Record Buffer Enable. Controls branch recording by the BRBE, and access to BRBE registers and instructions at EL2 and EL1.

SBRBE	Meaning
0b00	Direct accesses to BRBE registers and instructions, except when in EL3, generate a Trap exception to EL3. EL0, EL1, and EL2 are prohibited regions.
0b01	Direct accesses to BRBE registers and instructions in Secure state, except when in EL3, generate a Trap exception to EL3. EL0, EL1, and EL2 in Secure state are prohibited regions. This control does not cause any direct accesses to BRBE registers when not in Secure state to be trapped, and does not cause any Exception levels when not in Secure state to be a prohibited region.
0b10	Direct accesses to BRBE registers and instructions, except when in EL3, generate a Trap exception to EL3. This control does not cause any Exception levels to be prohibited regions.
0b11	This control does not cause any direct accesses to BRBE registers or instruction to be trapped, and does not cause any Exception levels to be a prohibited region.

The Branch Record Buffer registers trapped by this control are: [BRBCR_EL1](#), [BRBCR_EL2](#), [BRBCR_EL12](#), [BRBFCR_EL1](#), [BRBIDR0_EL1](#), [BRBINF<n>_EL1](#), [BRBINFINJ_EL1](#), [BRBSRC<n>_EL1](#), [BRBSRCINJ_EL1](#), [BRBTGT<n>_EL1](#), [BRBTGTINJ_EL1](#), and [BRBTS_EL1](#).

The Branch Record Buffer instructions trapped by this control are:

- [BRB IALL](#).
- [BRB INJ](#).

Note

If FEAT_BRBEv1p1 is not implemented, EL3 is a prohibited region.

If EL3 is not implemented then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BitBits [31:29]

Reserved, RES0.

PMSSE, bits [30:29]

When FEAT_PMUv3_SS is implemented:

Performance Monitors Snapshot Enable. Controls the generation of Capture events.

PMSSE	Meaning
0b00	Capture events are disabled.
0b01	Capture events are enabled and prohibited.
0b11	Capture events are enabled and allowed.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MTPME, bit [28]

When FEAT_MTPMU is implemented:

Multi-threaded PMU Enable. Enables use of the [PMEVTYPER<n>_ELO](#).MT bits.

MTPME	Meaning
0b0	FEAT_MTPMU is disabled. The Effective value of PMEVTYPER<n>_ELO .MT is zero.
0b1	PMEVTYPER<n>_ELO .MT bits not affected by this field.

If FEAT_MTPMU is disabled for any other PE in the system that has the same level 1 Affinity as the PE, it is IMPLEMENTATION DEFINED whether the PE behaves as if this field is 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 1.

Otherwise:

Reserved, RES0.

TDCC, bit [27]

When FEAT_FGT is implemented:

Trap DCC. Traps use of the Debug Comms Channel at EL2, EL1, and EL0 to EL3.

TDCC	Meaning
0b0	This control does not cause any register accesses to be trapped.
0b1	Accesses to the DCC registers at EL2, EL1, and EL0 generate a Trap exception to EL3, unless the access also generates a higher priority exception. Traps on the DCC data transfer registers are ignored when the PE is in Debug state.

The DCC registers trapped by this control are:

AArch64: [OSDTRRX_EL1](#), [OSDTRTX_EL1](#), [MDCCSR_EL0](#), [MDCCINT_EL1](#), and, when the PE is in Non-debug state, [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#).

AArch32: [DBGDTRRXint](#), [DBGDTRTXint](#), [DBGDSCRint](#), [DBGDCCINT](#), and, when the PE is in Non-debug state, [DBGDTRRXint](#) and [DBGDTRTXint](#).

The traps are reported with EC syndrome value:

- 0x05 for trapped AArch32 MRC and MCR accesses with coproc == 0b1110.
- 0x06 for trapped AArch32 LDC to [DBGDTRTXint](#) and STC from [DBGDTRRXint](#).
- 0x18 for trapped AArch64 MRS and MSR accesses.

When the PE is in Debug state, MDCR_EL3.TDCC does not trap any accesses to:

AArch64: [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#).

AArch32: [DBGDTRRXint](#) and [DBGDTRTXint](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSTBE, bit [26]

When FEAT_TRBE is implemented and FEAT_RME is implemented:

Non-secure Trace Buffer Extended. Together with MDCR_EL3.NSTB, controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

For a description of the values derived by evaluating NSTB and NSTBE together, see MDCR_EL3.NSTB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSTB, bits [25:24]

When FEAT_TRBE is implemented and FEAT_RME is implemented:

Non-secure Trace Buffer. Together with MDCR_EL3.NSTBE, controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

NSTBE	NSTB	Meaning
0b0	0b00	Secure state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Realm and Non-secure states. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When Secure state is not implemented, this encoding is reserved.
0b0	0b01	Secure state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Realm and Non-secure states. Accesses to Trace Buffer control registers at Realm and Non-secure EL2, and Realm and Non-secure EL1, generate Trap exceptions to EL3. When Secure state is not implemented, this encoding is reserved.
0b0	0b10	Non-secure state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Secure and Realm states. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b0	0b11	Non-secure state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Secure and Realm states. Accesses to Trace Buffer control registers at Secure and Realm EL2, and Secure and Realm EL1, generate Trap exceptions to EL3.
0b1	0b0x	Reserved
0b1	0b10	Realm state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Secure and Non-secure states. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b1	0b11	Realm state owns the Trace Buffer. When TraceBufferEnabled()==TRUE, tracing is prohibited in Secure and Non-secure states. Accesses to Trace Buffer control registers at Secure and Non-secure EL2, and Secure and Non-secure EL1, generate Trap exceptions to EL3.

NSTB	Meaning
0b00	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Reserved.
0b01	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Realm state generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Reserved.
0b10	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Trace Buffer owning Security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b11	When MDCR_EL3.NSTBE == 0b0: Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure and Realm state. Accesses to Trace Buffer control registers at EL2 and EL1 in Secure and Realm state generate Trap exceptions to EL3. When MDCR_EL3.NSTBE == 0b1: Trace Buffer owning Security state is Realm state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure and Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure and Secure state generate Trap exceptions to EL3.

The Trace Buffer control registers trapped by this control are: [TRBBASER_EL1](#), [TRBLIMITR_EL1](#), [TRBMAR_EL1](#), [TRBPTR_EL1](#), [TRBSR_EL1](#), and [TRBTRG_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_TRBE is implemented and FEAT_RME is not implemented:

Non-secure Trace Buffer. Controls the owning translation regime and accesses to Trace Buffer control registers from EL2 and EL1.

NSTB	Meaning
0b00	Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b01	Trace Buffer owning Security state is Secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Non-secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Non-secure state generate Trap exceptions to EL3.
0b10	Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b11	Trace Buffer owning Security state is Non-secure state. If TraceBufferEnabled() == TRUE, tracing is prohibited in Secure state. Accesses to Trace Buffer control registers at EL2 and EL1 in Secure state generate Trap exceptions to EL3.

The Trace Buffer control registers trapped by this control are: [TRBBASER_EL1](#), [TRBLIMITR_EL1](#), [TRBMAR_EL1](#), [TRBPTR_EL1](#), [TRBSR_EL1](#), and [TRBTRG_EL1](#).

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 1, then the Effective value of this field is 0b11.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0, then the Effective value of this field is 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SCCD, bit [23]

When FEAT_PMuV3p5 is implemented:

Secure Cycle Counter Disable. Prohibits [PMCCNTR_EL0](#) from counting in Secure state.

SCCD	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR_EL0 is prohibited in Secure state.

This field does not affect the CPU_CYCLES event or any other event that counts cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

ETAD, bit [22]

When FEAT_RME is implemented, external debugger access to the trace unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable. Together with MDCR_EL3.ETADE, controls access to trace unit registers by an external debugger.

ETADE	ETAD	Meaning
0b0	0b0	Access to trace unit registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to trace unit registers by an external debugger is permitted. Realm and Non-secure access to trace unit registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to trace unit registers by an external debugger is permitted. Secure and Non-secure access to trace unit registers by an external debugger is not permitted.
0b1	0b1	Root access to trace unit registers by an external debugger is permitted. Secure, Non-secure, and Realm access to trace unit registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When external debugger access to the trace unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable. Controls Non-secure access to trace unit registers by an external debugger.

ETAD	Meaning
0b0	Non-secure accesses from an external debugger to trace unit are allowed.
0b1	Non-secure accesses from an external debugger to some trace unit registers are prohibited. See individual registers for the effect of this field.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EPMAD, bit [21]

When FEAT_RME is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses:

External Performance Monitors Access Disable. Together with MDCR_EL3.EPMADE, controls access to Performance Monitor registers by an external debugger.

EPMADE	EPMAD	Meaning
0b0	0b0	Access to Performance Monitor registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to Performance Monitor registers by an external debugger is permitted. Realm and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Performance Monitor registers by an external debugger is permitted. Secure and Non-secure access to Performance Monitor registers by an external debugger is not permitted.
0b1	0b1	Root access to Performance Monitor registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Performance Monitor registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_Debugv8p4 is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses:

External Performance Monitors Non-secure Access Disable. Controls Non-secure access to Performance Monitor registers by an external debugger.

EPMAD	Meaning
0b0	Non-secure access to Performance Monitor registers from external debugger is permitted.
0b1	Non-secure access to Performance Monitor registers from external debugger is not permitted.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, then the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses:

External Performance Monitors Access Disable. Controls access to Performance Monitor registers by an external debugger.

EPMAD	Meaning
0b0	Access to Performance Monitor registers from external debugger is permitted.
0b1	Access to Performance Monitor registers from external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, then the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EDAD, bit [20]**When FEAT_RME is implemented:**

External Debug Access Disable. Together with MDCR_EL3.EDADE, controls access to breakpoint registers, watchpoint registers, and [OSLAR_EL1](#) by an external debugger.

EDADE	EDAD	Meaning
0b0	0b0	Access to Debug registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to Debug registers by an external debugger is permitted. Realm and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to Debug registers by an external debugger is permitted. Secure and Non-secure access to Debug registers by an external debugger is not permitted.
0b1	0b1	Root access to Debug registers by an external debugger is permitted. Secure, Non-secure, and Realm access to Debug registers by an external debugger is not permitted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_Debugv8p4 is implemented:

External Debug Non-secure Access Disable. Controls Non-secure access to breakpoint, watchpoint, and [OSLAR_EL1](#) registers by an external debugger.

EDAD	Meaning
0b0	Non-secure access to debug registers from external debugger is permitted.
0b1	Non-secure access to breakpoint and watchpoint registers, and OSLAR_EL1 from external debugger is not permitted.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_Debugv8p2 is implemented:

External Debug Access Disable. Controls access to breakpoint, watchpoint, and [OSLAR_EL1](#) registers by an external debugger.

EDAD	Meaning
0b0	Access to debug registers, and to OSLAR_EL1 from external debugger is permitted.
0b1	Access to breakpoint and watchpoint registers, and to OSLAR_EL1 from external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

External Debug Access disable. Controls access to breakpoint, watchpoint, and optionally [OSLAR_EL1](#) registers by an external debugger.

EDAD	Meaning
0b0	Access to debug registers from external debugger is permitted.
0b1	Access to breakpoint and watchpoint registers from an external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface. It is IMPLEMENTATION DEFINED whether access to the OSLAR_EL1 register from an external debugger is permitted or not permitted.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, then the Effective value of this field is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

TTRF, bit [19]**When FEAT_TRF is implemented:**

Trap Trace Filter controls. Traps use of the Trace Filter control registers at EL2 and EL1 to EL3.

The Trace Filter registers trapped by this control are:

- [TRFCR_EL2](#), [TRFCR_EL12](#), [TRFCR_EL1](#), reported using EC syndrome value 0x18.
- [HTRFCR](#) and [TRFCR](#), reported using EC syndrome value 0x03.

TTRF	Meaning
0b0	Accesses to Trace Filter registers at EL2 and EL1 are not affected by this bit.
0b1	Accesses to Trace Filter registers at EL2 and EL1 generate a Trap exception to EL3, unless the access generates a higher priority exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

STE, bit [18]**When FEAT_TRF is implemented and Secure state is implemented:**

Secure Trace enable. Enables tracing in Secure state.

STE	Meaning
0b0	Trace prohibited in Secure state unless overridden by the IMPLEMENTATION DEFINED authentication interface.
0b1	Trace in Secure state is not affected by this bit.

This bit also controls the level of authentication required by an external debugger to enable external tracing. See 'Register controls to enable self-hosted trace'.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0b0, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

SPME, bit [17]

When FEAT_PMUv3 is implemented and FEAT_PMUv3p7 is implemented:

Secure Performance Monitors Enable. Controls PMU event operation counting in Secure state and at EL3 when MDCR_EL3.MPMX is 0.EL3.

SPME	Meaning
0b0	When MDCR_EL3.MPMX == 0: Counters Event are counting is prohibited from counting in Secure state and at EL3.state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state and at EL3.state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
0b1	When MDCR_EL3.MPMX == 0: Counters Event are counting not affected by this mechanism.and PMCCNTR_EL0 are not affected by this mechanism.

When MDCR_EL3.MPMX is 0, this field affects the operation of all event counters affected in by Secure this state, field and are: if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 in Secure state.

When MDCR_EL3.MPMX is 1, this field affects the operation of event counters at EL3 only, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 at EL3 only. See MDCR_EL3.MPMX for more information.

- All event counters.
- If FEAT_PMUv3_ICNTR is implemented, the instruction counter, PMICNTR_EL0.
- If PMCR_EL0.DP is 1, the cycle counter, PMCCNTR_EL0.

When PMCR_EL0.DP is 0, PMCCNTR_EL0 is not affected by this field.

When MDCR_EL3.MPMX is 1, this field controls which event counters are affected by MDCR_EL3.MPMX at EL3. See MDCR_EL3.MPMX for more information.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_PMUv3 is implemented and FEAT_Debugv8p2 is implemented:

Secure Performance Monitors Enable. Controls PMU event operation counting in Secure state.

SPME	Meaning
0b0	When Event MDCR_EL3.MPMX counting == is 0: Counters are prohibited from counting in Secure state and at EL3.state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state and at EL3.state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
0b1	When Event MDCR_EL3.MPMX counting == 0: Counters are not affected by this mechanism.and PMCCNTR_EL0 are not affected by this mechanism.

This field affects the operation of all event counters in Secure state, and if PMCR_EL0.DP is 1, the operation of PMCCNTR_EL0 in Secure state. When PMCR_EL0.DP is 0, PMCCNTR_EL0 is not affected by this field.

If EL3 is not implemented and the Effective value of SCR_EL3.NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When FEAT_PMUv3 is implemented:

Secure Performance Monitors Enable. Controls **PMUevent operationcounting** in Secure state.

SPME	Meaning
0b0	If ExternalSecureNoninvasiveDebugEnabled() is FALSE, event counting is prohibited in Secure state, and if PMCR_EL0 .DP is 1, PMCCNTR_EL0 is disabled in Secure state.
0b1	Event counting and PMCCNTR_EL0 are not affected by this mechanism.

If ExternalSecureNoninvasiveDebugEnabled() is TRUE, the event counters and [PMCCNTR_EL0](#) are not affected by this field.

Otherwise, this field affects the operation of all event counters in Secure state, and if [PMCR_EL0](#).DP is 1, the operation of [PMCCNTR_EL0](#) in Secure state. When [PMCR_EL0](#).DP is 0, [PMCCNTR_EL0](#) is not affected by this field.

If EL3 is not implemented and the Effective value of [SCR_EL3](#).NS is 0, then the Effective value of this field is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

SDD, bit [16]**When Secure state is implemented:**

AArch64 Secure Self-hosted invasive debug disable. Disables Software debug exceptions in Secure state, other than Breakpoint Instruction exceptions.

SDD	Meaning
0b0	Debug exceptions in Secure state are not affected by this bit.
0b1	Debug exceptions, other than Breakpoint Instruction exceptions, are disabled from all Exception levels in Secure state.

The SDD bit is ignored unless both of the following are true:

- The PE is in Secure state.
- The Effective value of [SCR_EL3](#).RW is 0b1.

If Secure EL2 is implemented and enabled, and Secure EL1 is using AArch32, then:

- If debug exceptions from Secure EL1 are enabled, debug exceptions from Secure EL0 are also enabled.
- Otherwise, debug exceptions from Secure EL0 are enabled only if the value of [SDER32_EL3](#).SUIDEN is 0b1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SPD32, bits [15:14]**When EL1 is capable of using AArch32:**

AArch32 Secure self-hosted privileged debug. Enables or disables debug exceptions from Secure EL1 using AArch32, other than Breakpoint Instruction exceptions.

SPD32	Meaning
0b00	Legacy mode. Debug exceptions from Secure EL1 are enabled by the IMPLEMENTATION DEFINED authentication interface.
0b10	Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled.
0b11	Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled.

Other values are reserved, and have the CONSTRAINED UNPREDICTABLE behavior that they must have the same behavior as 0b00. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

This field has no effect on Breakpoint Instruction exceptions. These are always enabled.

This field is ignored unless both of the following are true:

- The PE is in Secure state.
- The Effective value of [SCR_EL3](#).RW is 0b0.

If Secure EL1 is using AArch32, then:

- If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled.
- Otherwise, debug exceptions from Secure EL0 are enabled only if the value of [SDER32_EL3](#).SUIDEN is 0b1.

If EL3 is not implemented and the Effective value of [SCR_EL3](#).NS is 0b0, then the Effective value of this field is 0b11.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSPB, bits [13:12]**When FEAT_SPE is implemented and FEAT_RME is implemented:**

Non-secure Profiling Buffer. Together with MDCR_EL3.NSPBE, controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control [registers from EL2 and EL1 registers.](#)

NSPBE	NSPB	Meaning
0b0	0b00	The Profiling Buffer uses Secure virtual addresses. Statistical Profiling is disabled in Realm and Non-secure states. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3. When Secure state is not implemented, this encoding is reserved.
0b0	0b01	The Profiling Buffer uses Secure virtual addresses. Statistical Profiling is disabled in Realm and Non-secure states. Accesses to Statistical Profiling and Profiling Buffer control registers at Realm and Non-secure EL2, and Realm and Non-secure EL1, generate Trap exceptions to EL3. When Secure state is not implemented, this encoding is reserved.
0b0	0b10	The Profiling Buffer uses Non-secure virtual addresses. Statistical Profiling is disabled in Secure and Realm states. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b0	0b11	The Profiling Buffer uses Non-secure virtual addresses. Statistical Profiling is disabled in Secure and Realm states. Accesses to Statistical Profiling and Profiling Buffer control registers at Secure and Realm EL2, and Secure and Realm EL1, generate Trap exceptions to EL3.
0b1	0b0x	Reserved
0b1	0b10	The Profiling Buffer uses Realm virtual addresses. Statistical Profiling is disabled in Secure and Non-secure states. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 generate Trap exceptions to EL3.
0b1	0b11	The Profiling Buffer uses Realm virtual addresses. Statistical Profiling is disabled in Secure and Non-secure states. Accesses to Statistical Profiling and Profiling Buffer control registers at Secure and Non-secure EL2, and Secure and Non-secure EL1, generate Trap exceptions to EL3.

NSPB	Meaning
0b00	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Reserved.
0b01	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Realm states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Reserved.
0b10	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Profiling Buffer uses Realm Virtual Addresses. Statistical Profiling enabled in Realm state and disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in all Security states generate Trap exceptions to EL3.
0b11	When MDCR_EL3.NSPBE == 0b0: Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure and Realm state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Secure and Realm states generate Trap exceptions to EL3. When MDCR_EL3.NSPBE == 0b1: Profiling Buffer uses Realm Virtual Addresses. Statistical Profiling enabled in Realm state and disabled in Non-secure and Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.

The Statistical Profiling and Profiling Buffer control registers trapped by this control are:

- [PMBLIMITR_EL1](#), [PMBPTR_EL1](#), [PMBSR_EL1](#), [PMSCR_EL1](#), [PMSCR_EL2](#), [PMSCR_EL12](#), [PMSEVFR_EL1](#), [PMSEFCR_EL1](#), [PMSICR_EL1](#), [PMSIDR_EL1](#), [PMSIRR_EL1](#), and [PMSLATFR_EL1](#).
- If FEAT_SPEv1p2 is implemented, [PMSNEVFR_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_SPE is implemented and FEAT_RME is not implemented:

Non-secure Profiling Buffer. Controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers.

NSPB	Meaning
0b00	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.
0b01	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure state generate Trap exceptions to EL3.
0b10	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Non-secure and Secure states generate Trap exceptions to EL3.
0b11	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer control registers at EL2 and EL1 in Secure state generate Trap exceptions to EL3.

The Statistical Profiling and Profiling Buffer control registers trapped by this control are:

- [PMBLIMITR_EL1](#), [PMBPTR_EL1](#), [PMBSR_EL1](#), [PMSCR_EL1](#), [PMSCR_EL2](#), [PMSCR_EL12](#), [PMSEVFR_EL1](#), [PMSFCR_EL1](#), [PMSICR_EL1](#), [PMSIDR_EL1](#), [PMSIRR_EL1](#), and [PMSLATFR_EL1](#).
- If FEAT_SPEv1p2 is implemented, [PMSNEVFR_EL1](#).

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 1, then the Effective value of this field is 0b11.

If EL3 is not implemented and the Effective value of [SCR_EL3.NS](#) is 0, then the Effective value of this field is 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSPBE, bit [11]

When FEAT_SPE is implemented and FEAT_RME is implemented:

Non-secure Profiling Buffer Extended. Together with MDCR_EL3.NSPB, controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control [registers from EL2 and EL1](#). ~~registers.~~

For a description of the values derived by evaluating NSPB and NSPBE together, see MDCR_EL3.NSPB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TDOSA, bit [10]

When FEAT_DoubleLock is implemented:

Trap debug OS-related register access. Traps EL2 and EL1 System register accesses to the powerdown debug registers to EL3.

Accesses to the registers are trapped as follows:

- Accesses from AArch64 state, [OSLAR_EL1](#), [OSLSR_EL1](#), [OSDLR_EL1](#), [DBGPRCR_EL1](#), and any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit, are trapped to EL3 and reported using EC syndrome value 0x18.
- Accesses using MCR or MRC to [DBGOSLAR](#), [DBGOSLSR](#), [DBGOSDLR](#), and [DBGPRCR](#), are trapped to EL3 and reported using EC syndrome value 0x05.
- Accesses to any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL2 and EL1 System register accesses to the powerdown debug registers are trapped to EL3, unless it is trapped by HDCR.TDOSA or MDCR_EL2.TDOSA .

Note

The powerdown debug registers are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Trap debug OS-related register access. Traps EL2 and EL1 System register accesses to the powerdown debug registers to EL3.

The following registers are affected by this trap:

- AArch64: [OSLAR_EL1](#), [OSLSR_EL1](#), and [DBGPRCR_EL1](#).
- AArch32: [DBGOSLAR](#), [DBGOSLSR](#), and [DBGPRCR](#).
- AArch64 and AArch32: Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.
- It is IMPLEMENTATION DEFINED whether accesses to [OSDLR_EL1](#) and [DBGOSDLR](#) are trapped.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL2 and EL1 System register accesses to the powerdown debug registers are trapped to EL3, unless it is trapped by HDCR.TDOSA or MDCR_EL2.TDOSA .

Note

The powerdown debug registers are not accessible at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDA, bit [9]

Trap Debug Access. Traps EL2, EL1, and EL0 System register accesses to those debug System registers that cannot be trapped using the MDCR_EL3.TDOSA field.

Accesses to the debug registers are trapped as follows:

- In AArch64 state, the following registers are trapped to EL3 and reported using EC syndrome value 0x18:
 - [DBGBVR<n>_EL1](#), [DBGBCR<n>_EL1](#), [DBGWVR<n>_EL1](#), [DBGWCR<n>_EL1](#), [DBGCLAIMSET_EL1](#), [DBGCLAIMCLR_EL1](#), [DBGAUTHSTATUS_EL1](#), [DBGVCR32_EL2](#).
 - AArch64: [MDCR_EL2](#), [MDRAR_EL1](#), [MDCCSR_EL0](#), [MDCCINT_EL1](#), [MDSCR_EL1](#), [OSDTRRX_EL1](#), [OSDTRTX_EL1](#), [OSECCR_EL1](#).
 - If FEAT_Debugv8p9 is implemented, [MDSELR_EL1](#).
- In AArch32 state, [SDER](#) is trapped to EL3 and reported using EC syndrome value 0x03.
- In AArch32 state, accesses using MCR or MRC to the following registers are reported using EC syndrome value 0x05, accesses using MCRR or MRRC are reported using EC syndrome value 0x0C:

- [HDCR](#), [DBGDRAR](#), [DBGDSAR](#), [DBGDIDR](#), [DBGDCCINT](#), [DBGWFEAR](#), [DBGVCR](#), [DBGBVR<n>](#), [DBGBCR<n>](#), [DBGBXVR<n>](#), [DBGWCR<n>](#), [DBGWVR<n>](#).
- [DBGCLAIMSET](#), [DBGCLAIMCLR](#), [DBGAUTHSTATUS](#), [DBGDEVID](#), [DBGDEVID1](#), [DBGDEVID2](#), [DBGOSECRR](#).
- In AArch32 state, STC accesses to [DBGDTRRXint](#) and LDC accesses to [DBGDTRTXint](#) are reported using EC syndrome value 0x06.
- When not in Debug state, the following registers are also trapped to EL3:
 - AArch64 accesses to [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#), reported using EC syndrome value 0x18.
 - AArch32 accesses using MCR or MRC to [DBGDTRRXint](#) and [DBGDTRTXint](#), reported using EC syndrome value 0x05.

TDA	Meaning
0b0	AccessesThis ofcontrol thedoes specifiednot debugcause Systemany registersinstructions areto notbe trapped by this mechanism.trapped.
0b1	AccessesEL0, ofEL1, and EL2 accesses to the specified debug Systemregisters, other than the registers atthat EL2,can EL1,be andtrapped EL0by MDCR_EL3.TDOSA, are trapped to EL3, unlessfrom theany instructionSecurity generatesstate and higherboth priorityExecution exception.states, unless it is trapped byDBGDSCRExt.UDCCdis, MDSCR_EL1.TDCC, HDCR.TDA or MDCR_EL2.TDA.

AArch64 accesses to [DBGDTR_EL0](#), [DBGDTRRX_EL0](#), and [DBGDTRTX_EL0](#), and AArch32 accesses using MCR or MRC to [DBGDTRRXint](#) and [DBGDTRTXint](#) are not trapped when the PE is in Debug state.

If 16 or fewer breakpoints and 16 or fewer watchpoints are implemented, and [MDSELR_EL1](#) is implemented as RAZ/WI, then it is IMPLEMENTATION DEFINED whether AArch64 accesses to [MDSELR_EL1](#) are trapped to EL3 when [MDCR_EL3.TDA](#) is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BitBits [8:7]

Reserved, RES0.

EnPM2, bit [7]

When FEAT_PMUv3p9 is implemented, or FEAT_SPMU is implemented or FEAT_SEBEP is implemented:

Enable access to additional PMU registers. When disabled, accesses to additional PMU registers generate a trap to EL3.

EnPM2	Meaning
0b0	Accesses of the specified additional PMU registers at EL2 and EL1 are trapped to EL3, unless the instruction generates a higher priority exception.
0b1	Accesses of the specified additional PMU registers are not trapped by this mechanism.

In AArch64 state, the instructions affected by this control are:

- If FEAT_EBEP is implemented, MRS and MSR accesses to [PMECR_EL1](#).
- If FEAT_SEBEP is implemented, MRS and MSR accesses to [PMIAR_EL1](#).
- If FEAT_PMUv3p9 is implemented, MRS and MSR accesses to [PMUACR_EL1](#).
- If FEAT_SPMU is implemented, MRS and MSR accesses to [SPMACCESSR_EL1](#), [SPMACCESSR_EL2](#), [SPMCFGFR_EL1](#), [SPMCGCR<n>_EL1](#), [SPMCNTENCLR_EL0](#), [SPMCNTENSET_EL0](#), [SPMCR_EL0](#), [SPMDEVAFF_EL1](#), [SPMDEVARCH_EL1](#), [SPMEVCNTR<n>_EL0](#), [SPMEVFILT2R<n>_EL0](#), [SPMEVFILTR<n>_EL0](#), [SPMEVTPER<n>_EL0](#), [SPMIIDR_EL1](#), [SPMINTENCLR_EL1](#), [SPMINTENSET_EL1](#), [SPMOVSLR_EL0](#), [SPMOVSSSET_EL0](#), [SPMSCR_EL1](#), and [SPMSELR_EL0](#).

Unless the instruction generates a higher priority exception, trapped instructions generate an exception to EL3.

Trapped instructions are reported using EC value 0x18.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TPM, bit [6]

When FEAT_PMUv3 is implemented:

Trap Performance Monitor register accesses. Accesses to all Performance Monitor registers from EL0, EL1, and EL2 to EL3, from any Security state and both Execution states are trapped as follows:

- In AArch64 state, accesses to the following registers are trapped to EL3 and are reported using EC syndrome value 0x18:
 - [PMCR_EL0](#), [PMCNTENSET_EL0](#), [PMCNTENCLR_EL0](#), [PMOVSCLR_EL0](#), [PMSWINC_EL0](#), [PMSELR_EL0](#), [PMCEID0_EL0](#), [PMCEID1_EL0](#), [PMCCNTR_EL0](#), [PMXEVTYPER_EL0](#), [PMXVCNTR_EL0](#), [PMUSERENR_EL0](#), [PMINTENSET_EL1](#), [PMINTENCLR_EL1](#), [PMOVSSET_EL0](#), [PMEVCNTR<n>_EL0](#), [PMEVTYPER<n>_EL0](#), [PMCCFILTR_EL0](#).
 - If FEAT_PMUv3p4 is implemented, [PMMIR_EL1](#).
- In AArch32 state, accesses using MCR or MRC to the following registers are reported using EC syndrome value 0x03, accesses using MCRR or MRRC are reported using EC syndrome value 0x04:
 - [PMCR](#), [PMCNTENSET](#), [PMCNTENCLR](#), [PMOVS](#), [PMSWINC](#), [PMSELR](#), [PMCEID0](#), [PMCEID1](#), [PMCCNTR](#), [PMXEVTYPER](#), [PMXVCNTR](#), [PMUSERENR](#), [PMINTENSET](#), [PMINTENCLR](#), [PMOVSSET](#), [PMEVCNTR<n>](#), [PMEVTYPER<n>](#), [PMCCFILTR](#).
 - If FEAT_PMUv3p1 is implemented, [PMCEID2](#), and [PMCEID3](#).
 - If FEAT_PMUv3p4 is implemented, [PMMIR](#).

TPM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL2, EL1, and EL0 System register accesses to all Performance Monitor registers are trapped to EL3, unless it is trapped by HDCR.TPM or MDCR_EL2.TPM .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [5]

Reserved, RES0.

EDADE, bit [4]

When FEAT_RME is implemented:

External Debug Access Disable Extended. Together with MDCR_EL3.EDAD, controls access to breakpoint registers, watchpoint registers, and [OSLAR_EL1](#) by an external debugger.

For a description of the values derived by evaluating EDAD and EDADE together, see MDCR_EL3.EDAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

ETADE, bit [3]

When FEAT_RME is implemented, external debugger access to the trace unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable Extended. Together with MDCR_EL3.ETAD, controls access to trace unit registers by an external debugger.

For a description of the values derived by evaluating ETAD and ETADE together, see MDCR_EL3.ETAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EPMAD, bit [2]

When FEAT_RME is implemented, FEAT_PMUv3 is implemented and the Performance Monitors Extension supports external debug interface accesses:

External Performance Monitors Access Disable Extended. Together with MDCR_EL3.EPMAD, controls access to Performance Monitor registers by an external debugger.

For a description of the values derived by evaluating EPMAD and EPMAD together, see MDCR_EL3.EPMAD.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [1]

Reserved, RES0.

RLTE, bit [0]

When FEAT_RME is implemented and FEAT_TRF is implemented:

Realm Trace enable. Enables tracing in Realm state.

RLTE	Meaning
0b0	Trace prohibited in Realm state, unless overridden by the IMPLEMENTATION DEFINED authentication interface.
0b1	Trace in Realm state is not affected by this bit.

This bit also controls the level of authentication that is required by an external debugger to enable external tracing.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Accessing MDCR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MDCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDCR_EL3;
```

MSR MDCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0011	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MDCR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MDRAR_EL1, Monitor Debug ROM Address Register

The MDRAR_EL1 characteristics are:

Purpose

Defines the base physical address of a 4KB-aligned memory-mapped debug component, usually a ROM table that locates and describes the memory-mapped debug components in the system. Armv8 deprecates any use of this register.

Configuration

AArch64 System register MDRAR_EL1 bits [63:0] are architecturally mapped to AArch32 System register [DBGDRAR\[63:0\]](#).

Attributes

MDRAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0								ROMADDR																								
ROMADDR												RES0																Valid				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:**56****52**]

Reserved, RES0.

ROMADDR, bits [**55****51**:12]

ROMADDR encoding when FEAT_D128FEAT_LPA is implemented and MDRAR_EL1.Valid != 0b00

43	42	41	40	39	38	37	36	35	34	33	32																																				
ROMADDR																																															
ROMADDR																																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
ROMADDR																																															

ROMADDR, bits [**43****39**:0]

Bits [**55****51**:12] of the ROM table physical address.

Bits [11:0] of the ROM table physical address are zero.

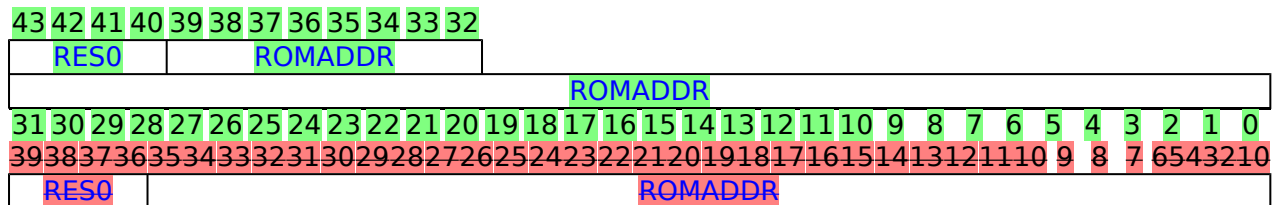
For implementations with fewer than **56****52** physical address bits, the corresponding upper bits of this field are RES0

In an implementation that includes EL3, ROMADDR is an address in Non-secure PA space. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure PA space. If FEAT_RME

is implemented, it is IMPLEMENTATION DEFINED whether the ROM table is also accessible in the Root or Realm PA spaces.

Arm strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system where the implementation only supports execution in AArch32 state.

ROMADDR encoding when FEAT_D128 is not implemented, FEAT_LPA is implemented and MDRAR_EL1.Valid != 0b00



Bits [43:39:40:36]

Reserved, RES0.

ROMADDR, bits [39:35:0]

Bits [51:39:12] of the ROM table physical address.

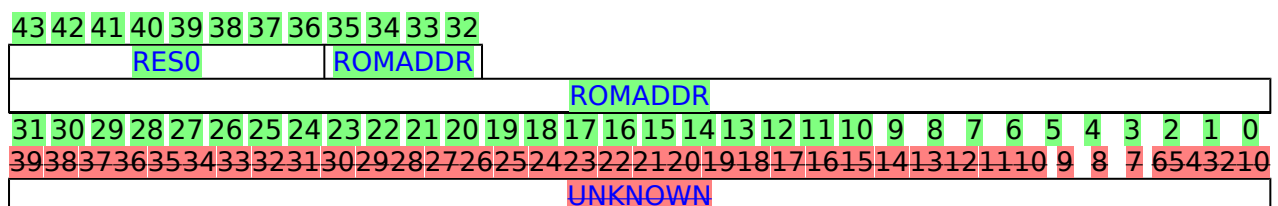
Bits [11:0] of the ROM table physical address are zero.

For implementations with fewer than 5248 physical address bits, the corresponding upper bits of this field are RES0

In an implementation that includes EL3, ROMADDR is an address in Non-secure PA space. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure PA space. If FEAT_RME is implemented, it is IMPLEMENTATION DEFINED whether the ROM table is also accessible in the Root or Realm PA spaces.

Arm strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system where the implementation only supports execution in AArch32 state.

ROMADDR encoding when FEAT_D128 is not implemented, FEAT_LPA is not implemented and MDRAR_EL1.Valid == 0b00



Bits [43:39:36:0]

Reserved, RES0UNKNOWN.

ROMADDR, bits [35:0]

Bits [39:12] of the ROM table physical address.

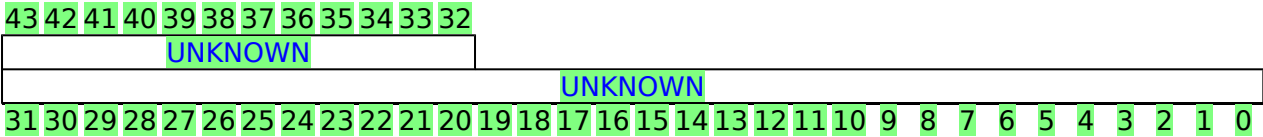
Bits [11:0] of the ROM table physical address are zero.

For implementations with fewer than 48 physical address bits, the corresponding upper bits of this field are RES0

In an implementation that includes EL3, ROMADDR is an address in Non-secure PA space. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure PA space. If FEAT_RME is implemented, it is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Root or Realm PA spaces.

Arm strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system where the implementation only supports execution in AArch32 state.

ROMADDR encoding when MDRAR_EL1.Valid == 0b00



Bits [43:0]

Reserved, UNKNOWN.

Bits [11:2]

Reserved, RES0.

Valid, bits [1:0]

This field indicates whether the ROM Table address is valid.

Valid	Meaning
0b00	ROM Table address is not valid. Software must ignore ROMADDR.
0b11	ROM Table address is valid.

Other values are reserved.

Arm recommends implementations set this field to zero.

Accessing MDRAR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MDRAR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDRA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDRAR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDRAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDRAR_EL1;

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MDSCR_EL1, Monitor Debug System Control Register

The MDSCR_EL1 characteristics are:

Purpose

Main control register for the debug implementation.

Configuration

AArch64 System register MDSCR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGDSCRext\[31:0\]](#).

AArch64 System register MDSCR_EL1 bit [15] is architecturally mapped to AArch32 System register [DBGDSCRint\[15\]](#).

AArch64 System register MDSCR_EL1 bit [12] is architecturally mapped to AArch32 System register [DBGDSCRint\[12\]](#).

AArch64 System register MDSCR_EL1 bits [5:2] are architecturally mapped to AArch32 System register [DBGDSCRint\[5:2\]](#).

AArch64 System register MDSCR_EL1 bit [40] is architecturally mapped to External register [EDSCR2\[8\]](#) when FEAT_Debugv8p9 is implemented.

Attributes

MDSCR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																																TTAEBWE	
TFORXfull		TXfull		RES0		RXOTXU		RES0		INTdis		TDARES0		SC2		RAZ/WI		MDEHDE		KDE		TDCC		RES0			ERR		RES0			SS	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:34]32]

Reserved, RES0.

TTA, bit [33]

When FEAT_TRBE_EXT is implemented or FEAT_ETEv1p3 is implemented:

Used for save/restore of [EDSCR2.TTA](#).

When [OSLSR_EL1.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1.OSLK](#) == 1, this bit holds the value of [EDSCR2.TTA](#). Reads and writes of this bit are indirect accesses to [EDSCR2.TTA](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

Otherwise:

Reserved, RES0.

EBWE, bit [32]**When FEAT_Debugv8p9 is implemented:**

Extended Breakpoint and Watchpoint Enable. Enables use of additional breakpoints or watchpoints.

EBWE	Meaning
0b0	Breakpoints and watchpoints above 15 are disabled, and the Effective value of MDSELR_EL1.BANK is zero.
0b1	Breakpoints and watchpoints above 15 are not affected by this field.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and MDSELR_EL1 is implemented as RAZ/WI.

This field is ignored by the PE and treated as zero when all of the following are true:

- Any of the following are true:
 - EL3 is implemented and MDCR_EL3.EBWE is 0.
 - EL2 is implemented and enabled in the current Security state, and MDCR_EL2.EBWE is 0.
- HaltOnBreakpointOrWatchpoint() is FALSE.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

TFO, bit [31]**When FEAT_TRF is implemented:**

Trace Filter override. Used for save/restore of EDSCR.TFO.

When OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.

When OSLSR_EL1.OSLK == 1, this bit holds the value of EDSCR.TFO. Reads and writes of this bit are indirect accesses to EDSCR.TFO.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

Otherwise:

Reserved, RES0.

RXfull, bit [30]

Used for save/restore of EDSCR.RXfull.

When OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).RXfull. Reads and writes of this bit are indirect accesses to [EDSCR](#).RXfull.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

TXfull, bit [29]

Used for save/restore of [EDSCR](#).TXfull.

When [OSLSR_EL1](#).OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).TXfull. Reads and writes of this bit are indirect accesses to [EDSCR](#).TXfull.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

Bit [28]

Reserved, RES0.

RXO, bit [27]

Used for save/restore of [EDSCR](#).RXO.

When [OSLSR_EL1](#).OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).RXO. Reads and writes of this bit are indirect accesses to [EDSCR](#).RXO.

When [OSLSR_EL1](#).OSLK == 1, if bits [27,6] of the value written to MDSCR_EL1 are {1,0}, that is, the RXO bit is 1 and the ERR bit is 0, the PE sets [EDSCR](#).{RXO,ERR} to UNKNOWN values.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

TXU, bit [26]

Used for save/restore of [EDSCR](#).TXU.

When [OSLSR_EL1](#).OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).TXU. Reads and writes of this bit are indirect accesses to [EDSCR](#).TXU.

When [OSLSR_EL1](#).OSLK == 1, if bits [26,6] of the value written to MDSCR_EL1 are {1,0}, that is, the TXU bit is 1 and the ERR bit is 0, the PE sets [EDSCR](#).{TXU,ERR} to UNKNOWN values.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When OSLSR_EL1.OSLK == 1, access to this field is **RW**.
- When OSLSR_EL1.OSLK == 0, access to this field is **RO**.

Bits [25:24]

Reserved, RES0.

INTdis, bits [23:22]

Used for save/restore of [EDSCR](#).INTdis.

When [OSLSR_EL1](#).OSLK == 0, and software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this field holds the value of [EDSCR](#).INTdis. Reads and writes of this field are indirect accesses to [EDSCR](#).INTdis.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [OSLSR_EL1](#).OSLK == 1, access to this field is **RW**.
- When [OSLSR_EL1](#).OSLK == 0, access to this field is **RO**.

TDA, bit [21]

Used for save/restore of [EDSCR](#).TDA.

When [OSLSR_EL1](#).OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).TDA. Reads and writes of this bit are indirect accesses to [EDSCR](#).TDA.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [OSLSR_EL1](#).OSLK == 1, access to this field is **RW**.
- When [OSLSR_EL1](#).OSLK == 0, access to this field is **RO**.

Bit [20]

Reserved, RES0.

SC2, bit [19]

When FEAT_PCSRv8 is implemented, FEAT_VHE is implemented and FEAT_PCSRv8p2 is not implemented:

Used for save/restore of [EDSCR](#).SC2.

When [OSLSR_EL1](#).OSLK == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1](#).OSLK == 1, this bit holds the value of [EDSCR](#).SC2. Reads and writes of this bit are indirect accesses to [EDSCR](#).SC2.

Accessing this field has the following behavior:

- When [OSLSR_EL1](#).OSLK == 1, access to this field is **RW**.
- When [OSLSR_EL1](#).OSLK == 0, access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [18:16]

Reserved, RAZ/WI.

Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

MDE, bit [15]

Monitor debug events. Enable Breakpoint, Watchpoint, and Vector Catch exceptions.

MDE	Meaning
0b0	Breakpoint, Watchpoint, and Vector Catch exceptions disabled.
0b1	Breakpoint, Watchpoint, and Vector Catch exceptions enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HDE, bit [14]

Used for save/restore of [EDSCR.HDE](#).

When [OSLSR_EL1.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1.OSLK](#) == 1, this bit holds the value of [EDSCR.HDE](#). Reads and writes of this bit are indirect accesses to [EDSCR.HDE](#).

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [OSLSR_EL1.OSLK](#) == 1, access to this field is **RW**.
- When [OSLSR_EL1.OSLK](#) == 0, access to this field is **RO**.

KDE, bit [13]

Local (kernel) debug enable. If [EL_D](#) is using AArch64, enable debug exceptions within [EL_D](#). Permitted values are:

KDE	Meaning
0b0	Debug exceptions, other than Breakpoint Instruction exceptions, disabled within EL_D .
0b1	All debug exceptions enabled within EL_D .

RES0 if [EL_D](#) is using AArch32.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TDCC, bit [12]

Traps EL0 accesses to the Debug Communication Channel (DCC) registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, from both Execution states, as follows:

- In AArch64 state, MRS or MSR accesses to the following DCC registers are trapped, reported using EC syndrome value 0x18:
 - [MDCCSR_EL0](#).
 - If not in Debug state, [DBGDTR_EL0](#), [DBGDTRTX_EL0](#), and [DBGDTRRX_EL0](#).
- In AArch32 state, MRC or MCR accesses to the following registers are trapped, reported using EC syndrome value 0x05.
 - [DBGDSCRint](#), [DBGDIDR](#), [DBGDSAR](#), [DBGDRAR](#).
 - If not in Debug state, [DBGDTRRXint](#), and [DBGDTRTXint](#).
- In AArch32 state, LDC access to [DBGDTRRXint](#) and STC access to [DBGDTRTXint](#) are trapped, reported using EC syndrome value 0x06.
- In AArch32 state, MRRC accesses to [DBGDSAR](#) and [DBGDRAR](#) are trapped, reported using EC syndrome value 0x0C.

TDCC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 using AArch64: EL0 accesses to the AArch64 DCC registers are trapped. EL0 using AArch32: EL0 accesses to the AArch32 DCC registers are trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:7]

Reserved, RES0.

ERR, bit [6]

Used for save/restore of [EDSCR.ERR](#).

When [OSLSR_EL1.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [OSLSR_EL1.OSLK](#) == 1, this bit holds the value of [EDSCR.ERR](#). Reads and writes of this bit are indirect accesses to [EDSCR.ERR](#).

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [OSLSR_EL1.OSLK](#) == 1, access to this field is **RW**.
- When [OSLSR_EL1.OSLK](#) == 0, access to this field is **RO**.

Bits [5:1]

Reserved, RES0.

SS, bit [0]

Software step control bit. If EL_D is using AArch64, enable Software step. Permitted values are:

SS	Meaning
0b0	Software step disabled
0b1	Software step enabled.

RES0 if EL_D is using AArch32.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MDSCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MDSCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.MDSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x158];
    else
        X[t, 64] = MDSCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDSCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDSCR_EL1;

```

MSR MDSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.MDSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x158] = X[t, 64];
    else
        MDSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MDSCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MDSELR_EL1, Breakpoint and Watchpoint Selection Register

The MDSELR_EL1 characteristics are:

Purpose

Selects the current breakpoints or watchpoints accessed by System register instructions.

Configuration

This register is present only when FEAT_Debugv8p9 is implemented. Otherwise, direct accesses to MDSELR_EL1 are UNDEFINED.

Attributes

MDSELR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																BANK				RES0											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:6]

Reserved, RES0.

BANK, bits [5:4]

When more than 16 breakpoints are implemented or more than 16 watchpoints are implemented:

Breakpoint and watchpoint bank select.

BANK	Meaning	Applies when
0b00	Select 0 to 15.	
0b01	Select 16 to 31.	
0b10	Select 32 to 47.	When at least 32 breakpoints or at least 32 watchpoints are implemented
0b11	Select 48 to 63.	When at least 48 breakpoints or at least 48 watchpoints are implemented

Each of the following register names accesses a register for breakpoint or watchpoint <n>, where n = UInt(MDSELR_EL1.BANK:m[3:0]):

- [DBGBCR<m>_EL1](#).
- [DBGBVR<m>_EL1](#).
- [DBGWCR<m>_EL1](#).
- [DBGWVR<m>_EL1](#).

This field is ignored by the PE and treated as zeros when the Effective value of [MDSCR_EL1.EBWE](#) is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [3:0]

Reserved, RES0.

Accessing MDSELR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MDSELR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDSELR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDSELR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDSELR_EL1;

```

MSR MDSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MDSELR_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

no old file

htmldiff from-

(new)

MECID_A0_EL2, Alternate MECID for EL2 and EL2&0 translation regimes

The MECID_A0_EL2 characteristics are:

Purpose

Alternate MECID for EL2 and EL2&0 accesses translated by [TTBR0_EL2](#).

Configuration

This register is present only when FEAT_MEC is implemented. Otherwise, direct accesses to MECID_A0_EL2 are UNDEFINED.

Attributes

MECID_A0_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																MECID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECID_A0_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECID_A0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1000	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2E8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MECID_A0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECID_A0_EL2;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

MECID_A1_EL2, Alternate MECID for EL2&0 translation regimes.

The MECID_A1_EL2 characteristics are:

Purpose

Alternate MECID for EL2&0 accesses translated by [TTBR1_EL2](#).

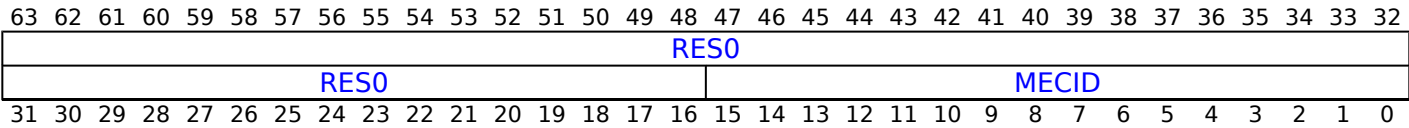
Configuration

This register is present only when FEAT_MEC is implemented. Otherwise, direct accesses to MECID_A1_EL2 are UNDEFINED.

Attributes

MECID_A1_EL2 is a 64-bit register.

Field descriptions



Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECID_A1_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECID_A1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1000	0b011


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2F8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MECID_A1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECID_A1_EL2;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

MECID_P0_EL2, Primary MECID for EL2 and EL2&0 translation regimes

The MECID_P0_EL2 characteristics are:

Purpose

Primary MECID for EL2 and EL2&0 accesses translated by [TTBR0_EL2](#).

Configuration

This register is present only when FEAT_MEC is implemented. Otherwise, direct accesses to MECID_P0_EL2 are UNDEFINED.

Attributes

MECID_P0_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																MECID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECID_P0_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECID_P0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2E0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MECID_P0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECID_P0_EL2;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

no old file

htmldiff from-

(new)

MECID_P1_EL2, Primary MECID for EL2&0 translation regimes

The MECID_P1_EL2 characteristics are:

Purpose

Primary MECID for EL2&0 accesses translated by [TTBR1_EL2](#).

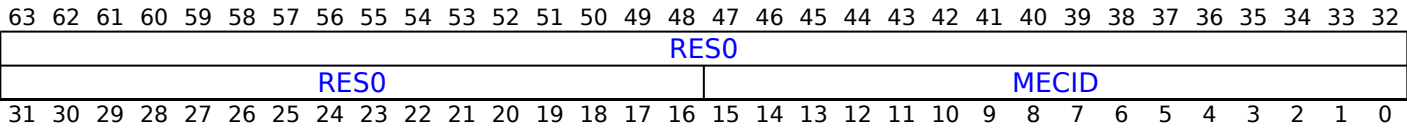
Configuration

This register is present only when FEAT_MEC is implemented. Otherwise, direct accesses to MECID_P1_EL2 are UNDEFINED.

Attributes

MECID_P1_EL2 is a 64-bit register.

Field descriptions



Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECID_P1_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECID_P1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1000	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2F0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MECID_P1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECID_P1_EL2;
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

no old file

htmldiff from-

(new)

MECID_RL_A_EL3, Realm PA space Alternate MECID for EL3 stage 1 translation regime

The MECID_RL_A_EL3 characteristics are:

Purpose

Realm PA space Alternate MECID for EL3 stage 1 translation regime.

Configuration

This register is present only when FEAT_MEC is implemented. Otherwise, direct accesses to MECID_RL_A_EL3 are UNDEFINED.

Attributes

MECID_RL_A_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																MECID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECID_RL_A_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECID_RL_A_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b1010	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECID_RL_A_EL3;
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

MECIDR_EL2, MEC Identification Register

The MECIDR_EL2 characteristics are:

Purpose

MEC identification register. Describes the supported MECID width by this PE.

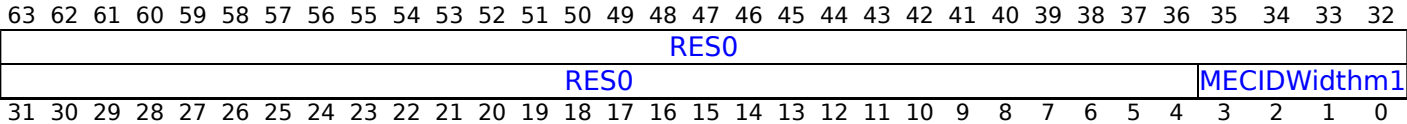
Configuration

This register is present only when FEAT_MEC is implemented and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to MECIDR_EL2 are UNDEFINED.

Attributes

MECIDR_EL2 is a 64-bit register.

Field descriptions



Bits [63:4]

Reserved, RES0.

MECIDWidthm1, bits [3:0]

The number of bits of MECID supported by the PE, minus 1.

The maximum permitted value is 0xF which indicates a MECID width of 16 bits and 2^16 MECIDs.

MECIDWidth is defined as MECIDWidthm1 + 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing MECIDR_EL2

For accesses from EL2 and EL3, this register is RO.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MECIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1000	0b111


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MECIDR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MECIDR_EL2;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

MFAR_EL3, PhysicalPA Fault Address Register (EL3)

The MFAR_EL3 characteristics are:

Purpose

RecordsHolds the faulting physical address for a Granule Protection Check, synchronous External Abort, or SError exceptionexceptions taken to EL3.

Configuration

This register is present only when FEAT_PFAR is implemented or FEAT_RME is implemented. Otherwise, direct accesses to MFAR_EL3 are UNDEFINED.

Attributes

MFAR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
NS	NSE	RES0						FPA[55:52]				FPA[51:48]				FPA[51:48]				FPA[47:12]				FPA[47:12]											
FPA[47:12]												RES0																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

An exception return at EL3 makes MFAR_EL3 UNKNOWN.

NS, bit [63]

Together with the NSE field, reports the physical address space of the access that triggered the Granule Protection Check exception.

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [62]

Together with the NS field, reports the physical address space of the access that triggered the Granule Protection Check exception.

For a description of the values derived by evaluating NS and NSE together, see MFAR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [61:56:52]

Reserved, RES0.

FPA[55:52], bits [55:52]**When FEAT_D128 is implemented:**

When FEAT_D128 is implemented, extension to FPA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FPA[51:48], bits [51:48]**When FEAT_LPA is implemented:**

When FEAT_LPA is implemented, extension to FPA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FPA[47:12], bits [47:12]

Bits [47:12] of the faulting physical address.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

When recording an address for a synchronous External Abort or SError exception, the recorded address can be any address within the same naturally-aligned fault granule as the faulting physical address, where the size of the fault granule is IMPLEMENTATION DEFINED, but must be no larger than:

- The size of the range of values permitted to be recorded in FAR_EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:0]

Reserved, RES0.

Accessing MFAR_EL3

MFAR_EL3 is not valid and reads UNKNOWN if ESR_EL3.PFV is recorded as 0 and ESR_EL3.EC is recorded indicating an Abort or SError exception.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MFAR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MFAR_EL3;

```

MSR MFAR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    MFAR_EL3 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MIDR_EL1, Main ID Register

The MIDR_EL1 characteristics are:

Purpose

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configuration

AArch64 System register MIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [MIDR\[31:0\]](#).

AArch64 System register MIDR_EL1 bits [31:0] are architecturally mapped to External register [MIDR_EL1\[31:0\]](#).

Attributes

MIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
Implementer								Variant				Architecture				PartNum												Revision			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

Implementer, bits [31:24]

The Implementer code. This field must hold an implementer code that has been assigned by Arm. Assigned codes include the following:

Implementer	Meaning
0x00	Reserved for software use.
0x41	Arm Limited.
0x42	Broadcom Corporation.
0x43	Cavium Inc.
0x44	Digital Equipment Corporation.
0x46	Fujitsu Ltd.
0x49	Infineon Technologies AG.
0x4D	Motorola or Freescale Semiconductor Inc.
0x4E	NVIDIA Corporation.
0x50	Applied Micro Circuits Corporation.
0x51	Qualcomm Inc.
0x56	Marvell International Ltd.
0x69	Intel Corporation.
0xC0	Ampere Computing.

Arm can assign codes that are not published in this manual. All values not assigned by Arm are reserved and must not be used.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Variant, bits [23:20]

Variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Architecture, bits [19:16]

Architecture version. Defined values are:

Architecture	Meaning
0b0001	Armv4.
0b0010	Armv4T.
0b0011	Armv5 (obsolete).
0b0100	Armv5T.
0b0101	Armv5TE.
0b0110	Armv5TEJ.
0b0111	Armv6.
0b1111	Architectural features are individually identified in the ID_* registers.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PartNum, bits [15:4]

Primary Part Number for the device.

On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [3:0]

Revision number for the device.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing MIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() then
        X[t, 64] = VPIDR_EL2;
    else
        X[t, 64] = MIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = MIDR_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MPIDR_EL1, Multiprocessor Affinity Register

The MPIDR_EL1 characteristics are:

Purpose

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configuration

AArch64 System register MPIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [MPIDR\[31:0\]](#).

In a uniprocessor system, Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

MPIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																								Aff3								
RES1	U	RES0						MT	Aff2								Aff1								Aff0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

Affinity level 3. See the description of Aff0 for more information.

Aff3 is not supported in AArch32 state.

Bit [31]

Reserved, RES1.

U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system.

U	Meaning
0b0	Processor is part of a multiprocessor system.
0b1	Processor is part of a uniprocessor system.

Bits [29:25]

Reserved, RES0.

MT, bit [24]

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels.

MT	Meaning
0b0	Performance of PEs with different affinity level 0 values, and the same values for affinity level 1 and higher, is largely independent.
0b1	Performance of PEs with different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.

Aff2, bits [23:16]

Affinity level 2. See the description of Aff0 for more information.

Aff1, bits [15:8]

Affinity level 1. See the description of Aff0 for more information.

Aff0, bits [7:0]

Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or [MPIDR_EL1](#).{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.

Accessing MPIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, MPIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b101

```
if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() then
        X[t, 64] = VMPIDR_EL2;
    else
        X[t, 64] = MPIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = MPIDR_EL1;
```

(old)	htmldiff from-	(new)
-------	----------------	-------

OSDLR_EL1, OS Double Lock Register

The OSDLR_EL1 characteristics are:

Purpose

Used to control the OS Double Lock.

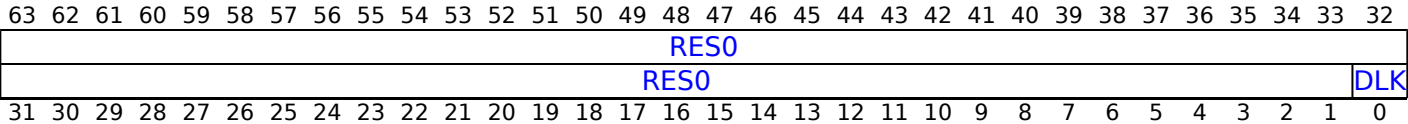
Configuration

AArch64 System register OSDLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGOSDLR\[31:0\]](#).

Attributes

OSDLR_EL1 is a 64-bit register.

Field descriptions



Bits [63:1]

Reserved, RES0.

DLK, bit [0] When FEAT_DoubleLock is implemented:

OS Double Lock control bit.

DLK	Meaning
0b0	OS Double Lock unlocked.
0b1	OS Double Lock locked, if DBGPRCR_EL1 .CORENPDRQ (Core no powerdown request) bit is set to 0 and the PE is in Non-debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RAZ/WI.

Accessing OSDLR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, OSDLR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_DoubleLock) && HDFGRTR_EL2.OSDLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' && (IsFeatureImplemented(FEAT_DoubleLock)
|| boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL2.TDOSA") then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = OSDLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = OSDLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = OSDLR_EL1;

```

MSR OSDLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_DoubleLock) && HDFGWTR_EL2.OSDLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' && (IsFeatureImplemented(FEAT_DoubleLock)
|| boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL2.TDOSA") then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            OSDLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' && (IsFeatureImplemented(FEAT_DoubleLock) ||
boolean IMPLEMENTATION_DEFINED "Trapped by MDCR_EL3.TDOSA") then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            OSDLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    OSDLR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

OSECCR_EL1, OS Lock Exception Catch Control Register

The OSECCR_EL1 characteristics are:

Purpose

Provides a mechanism for an operating system to access the contents of [EDECCR](#) that are otherwise invisible to software, so it can save/restore the contents of [EDECCR](#) over powerdown on behalf of the external debugger.

Configuration

AArch64 System register OSECCR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGOSECCR\[31:0\]](#).

AArch64 System register OSECCR_EL1 bits [31:0] are architecturally mapped to External register [EDECCR\[31:0\]](#).

If [OSLSR_EL1.OSLK](#) == 0, then OSECCR_EL1 returns an UNKNOWN value on reads and ignores writes.

Attributes

OSECCR_EL1 is a 64-bit register.

Field descriptions

When OSLSR_EL1.OSLK == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																EDECCR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

EDECCR, bits [31:0]

Used for save/restore to [EDECCR](#) over powerdown.

Reads or writes to this field are indirect accesses to [EDECCR](#).

Accessing OSECCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, OSECCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.OSECCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif OSLSR_EL1.OSLK == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = OSECCR_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif OSLSR_EL1.OSLK == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = OSECCR_EL1;
elseif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = OSECCR_EL1;

```

MSR OSECCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.OSECCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' then
        return;
    else
        OSECCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' then
        return;
    else
        OSECCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' then
        return;
    else
        OSECCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

OSLAR_EL1, OS Lock Access Register

The OSLAR_EL1 characteristics are:

Purpose

Used to lock or unlock the OS Lock.

Configuration

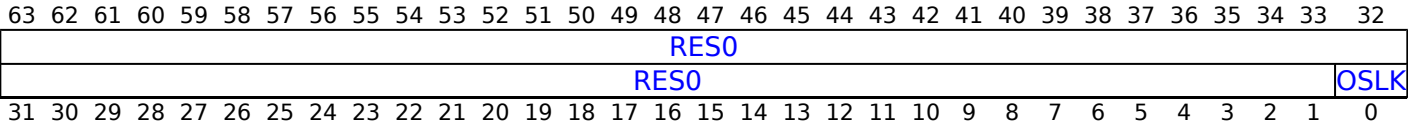
AArch64 System register OSLAR_EL1 bits [31:0] are architecturally mapped to External register [OSLAR_EL1\[31:0\]](#).

The OS Lock can also be locked or unlocked using [DBGOSLAR](#).

Attributes

OSLAR_EL1 is a 64-bit register.

Field descriptions



Bits [63:1]

Reserved, RES0.

OSLK, bit [0]

On writes to OSLAR_EL1, bit[0] is copied to the OS Lock.

Use [OSLSR_EL1](#).OSLK to check the current status of the lock.

Accessing OSLAR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MSR OSLAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0000	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.OSLAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        OSLAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        OSLAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    OSLAR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

OSLSR_EL1, OS Lock Status Register

The OSLSR_EL1 characteristics are:

Purpose

Provides the status of the OS Lock.

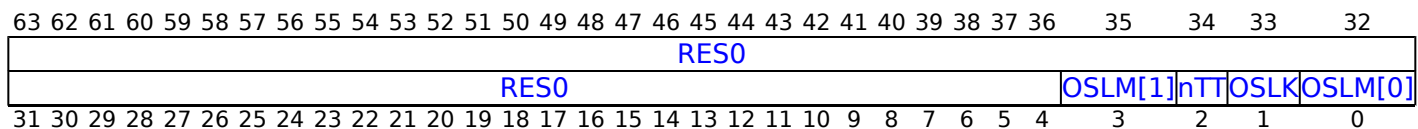
Configuration

AArch64 System register OSLSR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGOSLSR\[31:0\]](#).

Attributes

OSLSR_EL1 is a 64-bit register.

Field descriptions



Bits [63:4]

Reserved, RES0.

OSLM, bits [3, 0]

OS Lock model implemented. Identifies the form of OS save and restore mechanism implemented.

OSLM	Meaning
0b00	OS Lock not implemented.
0b10	OS Lock implemented.

All other values are reserved. In an Armv8 implementation the value 0b00 is not permitted.

The OSLM field is split as follows:

- OSLM[1] is OSLSR_EL1[3].
- OSLM[0] is OSLSR_EL1[0].

nTT, bit [2]

Not 32-bit access. This bit is always RAZ. It indicates that a 32-bit access is needed to write the key to the OS Lock Access Register.

OSLK, bit [1]

OS Lock Status.

OSLK	Meaning
0b0	OS Lock unlocked.
0b1	OS Lock locked.

The OS Lock is locked and unlocked by writing to the OS Lock Access Register.

The reset behavior of this field is:

- On a Cold reset, this field resets to 1.

Accessing OSLSR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, OSLSR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.OSLSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = OSLSR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = OSLSR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = OSLSR_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PAR_EL1, Physical Address Register

The PAR_EL1 characteristics are:

Purpose

Returns the output address (OA) from an Address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

Configuration

AArch64 System register PAR_EL1 bits [63:0] are architecturally mapped to AArch32 System register [PAR\[63:0\]](#).

AArch64 System register PAR_EL1 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Single stage AT Instructions (ATS1*) report their result using the 128-bit format of PAR_EL1 if the translation system that they target uses VMSAv9-128.

ATS12* Instructions report their result using the 128-bit format PAR_EL1 if either of the following is true:

- if stage 2 translations are enabled and the stage 2 translation system uses VMSAv9-128.
- if stage 2 translations are disabled and the stage 1 translation system uses VMSAv9-128.

Otherwise, 64-bit format of PAR_EL1 is used.

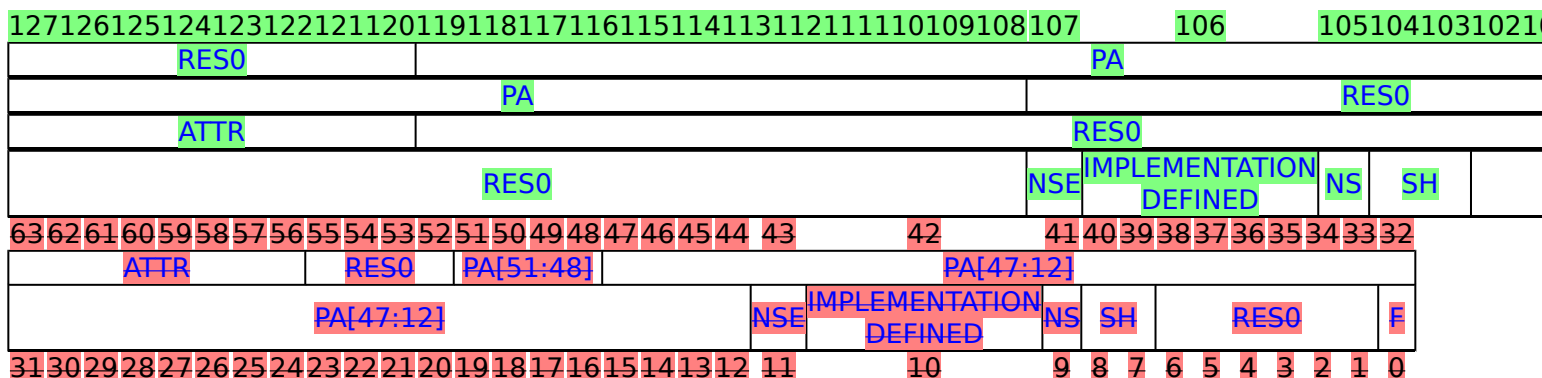
Attributes

PAR_EL1 is a: **64-bit register**.

- 128-bit register when FEAT_D128 is implemented, GetPAR_EL1_D128() == 1 and GetPAR_EL1_F() == 0
- 128-bit register when FEAT_D128 is implemented, GetPAR_EL1_D128() == 1 and GetPAR_EL1_F() == 1
- 128-bit register when FEAT_D128 is implemented, GetPAR_EL1_D128() == 0 and GetPAR_EL1_F() == 0
- 128-bit register when FEAT_D128 is implemented, GetPAR_EL1_D128() == 0 and GetPAR_EL1_F() == 1
- 64-bit register when FEAT_D128 is not implemented and GetPAR_EL1_F() == 0
- 64-bit register when FEAT_D128 is not implemented and GetPAR_EL1_F() == 1

Field descriptions

When **FEAT_D128 is implemented, GetPAR_EL1_D128() == 1 and GetPAR_EL1_F() == 0:**



This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

Bits [127:120]

Reserved, RES0.

PA, bits [119:76]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[55:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [75:65]

Reserved, RES0.

D128, bit [64]

Indicates if the PAR_EL1 uses the 128-bit format.

D128	Meaning
0b1	PAR_EL1 uses the 128-bit format. PAR_EL1[127:0] holds valid data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ATTR, bits [63:56]

Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in [MAIR_EL1](#), [MAIR_EL2](#), and [MAIR_EL3](#).

The value returned in this field can be the resulting attribute that is actually implemented by the implementation, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

Note

The attributes presented are consistent with the stages of translation applied in the address translation instruction. If the instruction performed a stage 1 translation only, the attributes are from the stage 1 translation. If the instruction performed a stage 1 and stage 2 translation, the attributes are from the combined stage 1 and stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [55:12]

Reserved, RES0.

NSE, bit [11]

When FEAT_RME is implemented:

Reports the NSE attribute for a translation table descriptor from the EL3 translation regime.

For a description of the values derived by evaluating NS and NSE together, see PAR_EL1.NS.

For a result from a Secure, Non-secure, or Realm translation regime, this bit is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

IMPLEMENTATION DEFINED, bit [10]

IMPLEMENTATION DEFINED.

NS, bit [9]

When FEAT_RME is implemented:

Non-secure. The NS attribute for a translation table descriptor from a Secure translation regime, a Realm translation regime, and the EL3 translation regime.

NS	Meaning
0b0	When NSE == 0: Secure. When NSE == 1: Root.
0b1	When NSE == 0: Non-secure. When NSE == 1: Realm.

For a result from a Secure translation regime, when SCR_EL3.EEL2 is 1, this bit reflects the Security state of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is unknown.

For a result from an S1E1 or S1E0 operation on the Realm EL1&0 translation regime, this bit is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_RME is not implemented:

Non-secure.

The NS attribute for a translation table descriptor from a Secure translation regime.

For a result from a Secure translation regime, when SCR_EL3.EEL2 is 1, this bit reflects the Security state of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SH, bits [8:7]

Shareability attribute, for the returned output address.

SH	Meaning
0b00	Non-shareable.
0b01	Outer Shareable.
0b10	Inner Shareable.
0b11	Reserved.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

Note

This field returns the value 0b10 for:

- Any type of Device memory.
- Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:1]

Reserved, RES0.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b0	Address translation completed successfully.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is implemented, GetPAR_EL1_D128() == 1 and GetPAR_EL1_F() == 1:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																RES0																																																																																																															
																RES0																																																																																																															
IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								RES0																																																																																																							
RES0																DirtyBitOverlay								TopLevel								AssuredOnly								RES1								RES0																																																																															

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

Bits [127:65]

Reserved, RES0.

D128, bit [64]

Indicates if the PAR_EL1 uses the 128-bit format.

D128	Meaning
0b1	PAR_EL1 uses the 128-bit format. PAR_EL1[127:0] holds valid data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [63:56]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [55:52]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [51:48]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [47:16]

Reserved, RES0.

DirtyBit, bit [15]

When FEAT_S1PIE is implemented or FEAT_S2PIE is implemented:

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, this field holds information about the fault.

DirtyBit	Meaning
0b0	The Permission Fault is not due to nDirty State or Dirty State.
0b1	The Permission Fault is due to nDirty State or Dirty State.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [14]

When FEAT_S1POE is implemented or FEAT_S2POE is implemented:

Overlay flag. If a memory access generates a Data Abort for a Permission fault, this field holds information about the fault.

Overlay	Meaning
0b0	The Data Abort is due to Base Permissions.
0b1	The Data Abort is due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TopLevel, bit [13]

When FEAT_THE is implemented:

Fault due to TopLevel. Indicates if the fault was due to TopLevel.

TopLevel	Meaning
0b0	Fault is not due to TopLevel.
0b1	Fault is due to TopLevel.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AssuredOnly, bit [12]

When FEAT_THE is implemented:

AssuredOnly flag.

If a memory access generates a Stage 2 Data Abort, this field holds information about the fault.

AssuredOnly	Meaning
0b0	The Data Abort is not due to AssuredOnly.
0b1	The Data Abort is due to AssuredOnly.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [11]

Reserved, RES1.

Bit [10]

Reserved, RES0.

S, bit [9]

Indicates the translation stage at which the translation aborted:

S	Meaning
0b0	Translation aborted because of a fault in the stage 1 translation.
0b1	Translation aborted because of a fault in the stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [8]

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

FST, bits [6:1]

Fault status code, as shown in the Data Abort ESR encoding.

FST	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented

0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b111101	Section Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32
0b111110	Page Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b1	Address translation aborted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is implemented, GetPAR_EL1_D128() == 0 and GetPAR_EL1_F() == 0:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107		106		105	104	103	102	101	
RES0																													
RES0																													
ATTR						RES0						PA[51:48]						PA[47:12]											
PA[47:12]																		NSE		IMPLEMENTATION DEFINED				NS		SH			

This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR_EL1 can return a value that indicates the resulting attributes, rather than the values that appear in the Translation table descriptors. More precisely:

- The PAR_EL1.{ATTR, SH} fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the Translation table descriptors.
- See the PAR_EL1.NS bit description for constraints on the value it returns.

Bits [127:65]

Reserved, RES0.

D128, bit [64]

Indicates if the PAR_EL1 uses the 128-bit format.

D128	Meaning
0b1	PAR_EL1 uses the 128-bit format. PAR_EL1[127:0] holds valid data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ATTR, bits [63:56]

Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in [MAIR_EL1](#), [MAIR_EL2](#), and [MAIR_EL3](#).

The value returned in this field can be the resulting attribute that is actually implemented by the implementation, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the Translation table descriptor.

Note

The attributes presented are consistent with the stages of translation applied in the address translation instruction. If the instruction performed a stage 1 translation only, the attributes are from the stage 1 translation. If the instruction performed a stage 1 and stage 2 translation, the attributes are from the combined stage 1 and stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [55:52]

Reserved, RES0.

PA[51:48], bits [51:48]**When FEAT_LPA is implemented:**

Extension to PA[47:12]. For more information, see PA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PA[47:12], bits [47:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:12].

When FEAT_LPA is implemented and 52-bit addresses are in use, PA[51:48] forms the upper part of the address value. Otherwise, when 52-bit addresses are not in use, PA[51:48] is RES0.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [11]**When FEAT_RME is implemented:**

Reports the NSE attribute for a translation table entry from the EL3 translation regime.

For a description of the values derived by evaluating NS and NSE together, see PAR_EL1.NS.

For a result from a Secure, Non-secure, or Realm translation regime, this bit is UNKNOWN.

Otherwise:

Reserved, RES1.

IMPLEMENTATION DEFINED, bit [10]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NS, bit [9]**When FEAT_RME is implemented:**

Non-secure. The NS attribute for a translation table entry from a Secure translation regime, a Realm translation regime, and the EL3 translation regime.

For a result from an EL3 translation regime, NS and NSE are evaluated together to report the physical address space:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

For a result from a Secure translation regime, when SCR_EL3.EEL2 is 1, this bit distinguishes between the Secure state and state Non-secure of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

For a result from an S1E1 or S1E0 operation on the Realm EL1&0 translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, when [SCR_EL3.EEL2](#) is 1, this bit distinguishes between the Secure Security and state Non-secure of the intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH, bits [8:7]

Shareability attribute, for the returned output address.

SH	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

The value 0b01 is reserved.

Note

This field returns the value 0b10 for:

- Any type of Device memory.
- Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the Translation table descriptor.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:1]

Reserved, RES0.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b0	Address translation completed successfully.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is implemented, GetPAR_EL1_D128() == 0 and GetPAR_EL1_F()PAR_EL1.F == 1:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																RES0																																																																																																															
																RES0																																																																																																															
IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED														RES0																																																																																																					
																RES0										DirtyBit				Overlay				TopLevel				AssuredOnly				RES1		RES0																																																																																			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																
IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED														RES0																																																																																																					
																RES0										RES1				RES0				S				PTW				RES0				FST				F																																																																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

Bits [127:65]

Reserved, RES0.

D128, bit [64]

Indicates if the PAR_EL1 uses the 128-bit format.

D128	Meaning
0b1	PAR_EL1 uses the 128-bit format. PAR_EL1[127:0] holds valid data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [63:56]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [55:52]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [51:48]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [47:16:12]

Reserved, RES0.

DirtyBit, bit [15]**When FEAT_S1PIE is implemented or FEAT_S2PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, this field holds information about the fault.

DirtyBit	Meaning
0b0	The Permission Fault is not due to nDirty State or Dirty State.
0b1	The Permission Fault is due to nDirty State or Dirty State.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [14]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag. If a memory access generates a Data Abort for a Permission fault, this field holds information about the fault.

Overlay	Meaning
0b0	The Data Abort is due to Base Permissions.
0b1	The Data Abort is due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TopLevel, bit [13]**When FEAT_THE is implemented:**

Fault due to TopLevel. Indicates if the fault was due to TopLevel.

TopLevel	Meaning
0b1	Fault is due to TopLevel.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AssuredOnly, bit [12]**When FEAT_TTE is implemented:**

AssuredOnly flag.

If a memory access generates a Stage 2 Data Abort, this field holds information about the fault.

AssuredOnly	Meaning
0b0	The Data Abort is not due to AssuredOnly.
0b1	The Data Abort is due to AssuredOnly.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [11]

Reserved, RES1.

Bit [10]

Reserved, RES0.

S, bit [9]

Indicates the translation stage at which the translation aborted:

S	Meaning
0b0	Translation aborted because of a fault in the stage 1 translation.
0b1	Translation aborted because of a fault in the stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [8]

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

FST, bits [6:1]

Fault status code, as shown in the Data Abort exception ESR encoding.

FST	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented

0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b111101	Section Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32
0b111110	Page Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b1	Address translation aborted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PAR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PAR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PAR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = PAR_EL1;

```

MSR PAR_EL1, <Xt>

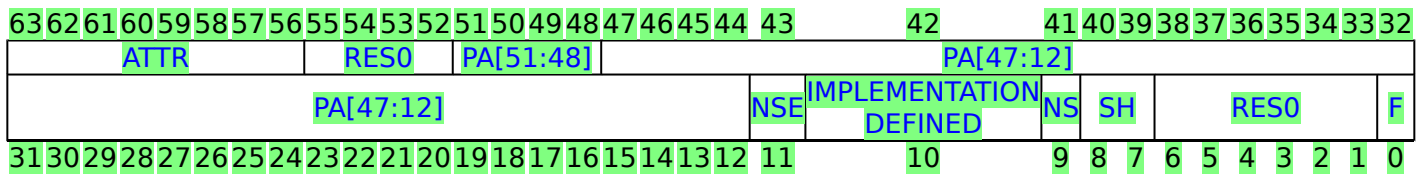
op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PAR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PAR_EL1 = X[t, 64];
elseif PSTATE.EL == EL3 then
    PAR_EL1 = X[t, 64];

```

When FEAT_D128 is not implemented and GetPAR_EL1_F() == 0:



This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR_EL1 can return a value that indicates the resulting attributes, rather than the values that appear in the Translation table descriptors. More precisely:

- The PAR_EL1.{ATTR, SH} fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the Translation table descriptors.
- See the PAR_EL1.NS bit description for constraints on the value it returns.

ATTR, bits [63:56]

Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in MAIR EL1, MAIR EL2, and MAIR EL3.

If FEAT_MTE_PERM is implemented and the instruction performed a stage 2 translation, the following additional encoding is defined:

ATTR	Meaning
0b11100000	Tagged NoTagAccess Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Note: This encoding in MAIR ELx is Reserved

The value returned in this field can be the resulting attribute that is actually implemented by the implementation, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the Translation table descriptor.

Note

The attributes presented are consistent with the stages of translation applied in the address translation instruction. If the instruction performed a stage 1 translation only, the attributes are from the stage 1 translation. If the instruction performed a stage 1 and stage 2 translation, the attributes are from the combined stage 1 and stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [55:52]

Reserved, RES0.

PA[51:48], bits [51:48]

When FEAT_LPA is implemented:

Extension to PA[47:12]. For more information, see PA[47:12].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PA[47:12], bits [47:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:12].

When FEAT_LPA is implemented and 52-bit addresses are in use, PA[51:48] forms the upper part of the address value. Otherwise, when 52-bit addresses are not in use, PA[51:48] is RES0.

For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSE, bit [11]

When FEAT_RME is implemented:

Reports the NSE attribute for a translation table entry from the EL3 translation regime.

For a description of the values derived by evaluating NS and NSE together, see PAR_EL1.NS.

For a result from a Secure, Non-secure, or Realm translation regime, this bit is UNKNOWN.

Otherwise:

Reserved, RES1.

IMPLEMENTATION DEFINED, bit [10]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NS, bit [9]**When FEAT_RME is implemented:**

Non-secure. The NS attribute for a translation table entry from a Secure translation regime, a Realm translation regime, and the EL3 translation regime.

For a result from an EL3 translation regime, NS and NSE are evaluated together to report the physical address space:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

For a result from a Secure translation regime, when [SCR_EL3.EEL2](#) is 1, this bit distinguishes between the Secure and Non-secure intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

For a result from an S1E1 or S1E0 operation on the Realm EL1&0 translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, when [SCR_EL3.EEL2](#) is 1, this bit distinguishes between the Secure and Non-secure intermediate physical address space of the translation for the instructions:

- In AArch64 state: [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#), [AT S1E0R](#), and [AT S1E0W](#).
- In AArch32 state: [ATS1CPR](#), [ATS1CPW](#), [ATS1CPRP](#), [ATS1CPWP](#), [ATS1CUR](#), and [ATS1CUW](#).

Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH, bits [8:7]

Shareability attribute, for the returned output address.

SH	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

The value 0b01 is reserved.

Note

This field returns the value 0b10 for:

- Any type of Device memory.
- Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the Translation table descriptor.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [6:1]

Reserved, RES0.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b0	Address translation completed successfully.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is not implemented and GetPAR_EL1_F() == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39								
IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								RES0								
RES0																DirtyBit	Overlay	TopLevel	AssuredOnly	RES1	RES0	S	PTW	RES								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7								

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

IMPLEMENTATION DEFINED, bits [63:56]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [55:52]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IMPLEMENTATION DEFINED, bits [51:48]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [47:16]

Reserved, RES0.

DirtyBit, bit [15]**When FEAT_S1PIE is implemented or FEAT_S2PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, this field holds information about the fault.

DirtyBit	Meaning
0b0	The Permission Fault is not due to nDirty State or Dirty State.
0b1	The Permission Fault is due to nDirty State or Dirty State.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [14]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag. If a memory access generates a Data Abort for a Permission fault, this field holds information about the fault.

Overlay	Meaning
0b0	The Data Abort is due to Base Permissions.
0b1	The Data Abort is due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TopLevel, bit [13]**When FEAT_THE is implemented:**

Fault due to TopLevel. Indicates if the fault was due to TopLevel.

TopLevel	Meaning
0b1	Fault is due to TopLevel.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AssuredOnly, bit [12]

When FEAT_TTE is implemented:

AssuredOnly flag.

If a memory access generates a Stage 2 Data Abort, this field holds information about the fault.

AssuredOnly	Meaning
0b0	The Data Abort is not due to AssuredOnly.
0b1	The Data Abort is due to AssuredOnly.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [11]

Reserved, RES1.

Bit [10]

Reserved, RES0.

S, bit [9]

Indicates the translation stage at which the translation aborted:

S	Meaning
0b0	Translation aborted because of a fault in the stage 1 translation.
0b1	Translation aborted because of a fault in the stage 2 translation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PTW, bit [8]

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [7]

Reserved, RES0.

FST, bits [6:1]

Fault status code, as shown in the Data Abort exception ESR encoding.

FST	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented

0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
0b111101	Section Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32
0b111110	Page Domain fault, from an AArch32 stage 1 EL1&0 translation regime using Short-descriptor translation table format.	When EL1 is capable of using AArch32

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0b1	Address translation aborted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PAR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PAR_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PAR_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PAR_EL1<63:0>;

```

MSR PAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PAR_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    PAR_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    PAR_EL1<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, PAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (PAR_EL1<127:64>, PAR_EL1<63:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (PAR_EL1<127:64>, PAR_EL1<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (PAR_EL1<127:64>, PAR_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR PAR_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (PAR_EL1<127:64>, PAR_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (PAR_EL1<127:64>, PAR_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
elsif PSTATE.EL == EL3 then
    (PAR_EL1<127:64>, PAR_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PFAR_EL1, Physical Fault Address Register (EL1)

The PFAR_EL1 characteristics are:

Purpose

Records the faulting physical address for a synchronous External Abort, or SError exception taken to EL1.

Configuration

This register is present only when FEAT_PFAR is implemented. Otherwise, direct accesses to PFAR_EL1 are UNDEFINED.

Attributes

PFAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NS	NSE	RESO										PA																			
PA																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NS, bit [63]

When FEAT_RME is implemented:

Together with the NSE field, reports the physical address space of the access that triggered the exception.

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When EL3 is implemented:

Non-secure. Reports the physical address space of the access that triggered the exception.

NS	Meaning
0b0	Secure physical address space.
0b1	Non-secure physical address space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSE, bit [62]**When FEAT_RME is implemented:**

Together with the NS field, reports the physical address space of the access that triggered the exception.

For a description of the values derived by evaluating NS and NSE together, see MFAR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [61:52]

Reserved, RES0.

PA, bits [51:0]

Physical Address.

For implementations with fewer than 52 physical address bits, the corresponding upper bits in this field are RES0.

When recording an address for a synchronous External Abort or SError exception, the recorded address can be any address within the same naturally-aligned fault granule as the faulting physical address, where the size of the fault granule is IMPLEMENTATION DEFINED, but must be no larger than:

- The size of the range of values permitted to be recorded in [FAR_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PFAR_EL1

PFAR_EL1 is not valid and reads UNKNOWN if [ESR_EL1](#).PFV is recorded as 0.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PFAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGTR2_EL2.nPFAR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x2D0];
    else
        X[t, 64] = PFAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = PFAR_EL2;
    else
        X[t, 64] = PFAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PFAR_EL1;

```

MSR PFAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGWTR2_EL2.nPFAR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x2D0] = X[t, 64];
    else
        PFAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        PFAR_EL2 = X[t, 64];
    else
        PFAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PFAR_EL1 = X[t, 64];

```

MRS <Xt>, PFAR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x2D0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = PFAR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = PFAR_EL1;
    else
        UNDEFINED;

```

MSR PFAR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x2D0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        PFAR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        PFAR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PFAR_EL2, Physical Fault Address Register (EL2)

The PFAR_EL2 characteristics are:

Purpose

Records the faulting physical address for a synchronous External Abort, or SError exception taken to EL2.

Configuration

This register is present only when FEAT_PFAR is implemented. Otherwise, direct accesses to PFAR_EL2 are UNDEFINED.

Attributes

PFAR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NS	NSE	RESO										PA																			
PA																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NS, bit [63]

When FEAT_RME is implemented:

Together with the NSE field, reports the physical address space of the access that triggered the exception.

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When EL3 is implemented:

Non-secure. Reports the physical address space of the access that triggered the exception.

NS	Meaning
0b0	Secure physical address space.
0b1	Non-secure physical address space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSE, bit [62]**When FEAT_RME is implemented:**

Together with the NS field, reports the physical address space of the access that triggered the exception.

For a description of the values derived by evaluating NS and NSE together, see MFAR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [61:52]

Reserved, RES0.

PA, bits [51:0]

Physical Address.

For implementations with fewer than 52 physical address bits, the corresponding upper bits in this field are RES0.

When recording an address for a synchronous External Abort or SError exception, the recorded address can be any address within the same naturally-aligned fault granule as the faulting physical address, where the size of the fault granule is IMPLEMENTATION DEFINED, but must be no larger than:

- The size of the range of values permitted to be recorded in [FAR_EL2](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PFAR_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name PFAR_EL2 or PFAR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

PFAR_EL2 is not valid and reads UNKNOWN if [ESR_EL2.PFV](#) is recorded as 0.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PFAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b101


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PFAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PFAR_EL2;

```

MSR PFAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    PFAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PFAR_EL2 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PIR_EL1, Permission Indirection Register 1 (EL1)

The PIR_EL1 characteristics are:

Purpose

Stage 1 Permission Indirection Register for privileged access of the EL1&0 translation regime.

Configuration

This register is present only when FEAT_S1PIE is implemented. Otherwise, direct accesses to PIR_EL1 are UNDEFINED.

Attributes

PIR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents Stage 1 Base Permissions.

Perm<m>	Meaning
0b0000	No access, Overlay applied.
0b0001	Read, Overlay applied.
0b0010	Execute, Overlay applied.
0b0011	Read and Execute, Overlay applied.
0b0100	Reserved - treated as No access, Overlay applied.
0b0101	Read and Write, Overlay applied.
0b0110	Read, Write and Execute, Overlay applied.
0b0111	Read, Write and Execute, Overlay applied.
0b1000	Read, Overlay not applied.
0b1001	Read, GCS Read and GCS Write, Overlay not applied.
0b1010	Read and Execute, Overlay not applied.
0b1011	Reserved - treated as No access, Overlay not applied.
0b1100	Read and Write, Overlay not applied.
0b1101	Reserved - treated as No access, Overlay not applied.
0b1110	Read, Write and Execute, Overlay not applied.
0b1111	Reserved - treated as No access, Overlay not applied.

This field is permitted to be cached in a TLB.

When Stage 1 Indirect Permission mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PIR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPIR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x2A0];
    else
        X[t, 64] = PIR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PIR_EL2;
    else
        X[t, 64] = PIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIR_EL1;

```

MSR PIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPIR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x2A0] = X[t, 64];
    else
        PIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        PIR_EL2 = X[t, 64];
    else
        PIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIR_EL1 = X[t, 64];

```

MRS <Xt>, PIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x2A0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = PIR_EL1;
    else
        UNDEFINED;

```

MSR PIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x2A0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        PIR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PIR_EL2, Permission Indirection Register 2 (EL2)

The PIR_EL2 characteristics are:

Purpose

Stage 1 Permission Indirection Register for privileged access of the EL2 or EL2&0 translation regime.

Configuration

This register is present only when FEAT_S1PIE is implemented. Otherwise, direct accesses to PIR_EL2 are UNDEFINED.

Attributes

PIR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents Stage 1 Base Permissions.

Perm<m>	Meaning
0b0000	No access, Overlay applied.
0b0001	Read, Overlay applied.
0b0010	Execute, Overlay applied.
0b0011	Read and Execute, Overlay applied.
0b0100	Reserved - treated as No access, Overlay applied.
0b0101	Read and Write, Overlay applied.
0b0110	Read, Write and Execute, Overlay applied.
0b0111	Read, Write and Execute, Overlay applied.
0b1000	Read, Overlay not applied.
0b1001	Read, GCS Read and GCS Write, Overlay not applied.
0b1010	Read and Execute, Overlay not applied.
0b1011	Reserved - treated as No access, Overlay not applied.
0b1100	Read and Write, Overlay not applied.
0b1101	Reserved - treated as No access, Overlay not applied.
0b1110	Read, Write and Execute, Overlay not applied.
0b1111	Reserved - treated as No access, Overlay not applied.

This field is permitted to be cached in a TLB.

When Stage 1 Indirect Permission mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PIR_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name PIR_EL2 or [PIR_EL1](#) are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIR_EL2;

```

MSR PIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIR_EL2 = X[t, 64];

```


MRS <Xt>, PIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPIR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x2A0];
    else
        X[t, 64] = PIR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PIR_EL2;
    else
        X[t, 64] = PIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIR_EL1;

```

MSR PIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPIR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x2A0] = X[t, 64];
    else
        PIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        PIR_EL2 = X[t, 64];
    else
        PIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIR_EL1 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PIR_EL3, Permission Indirection Register 3 (EL3)

The PIR_EL3 characteristics are:

Purpose

Stage 1 Permission Indirection Register for privileged access of the EL3 translation regime.

Configuration

This register is present only when FEAT_S1PIE is implemented. Otherwise, direct accesses to PIR_EL3 are UNDEFINED.

Attributes

PIR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15				Perm14				Perm13				Perm12				Perm11				Perm10				Perm9				Perm8			
Perm7				Perm6				Perm5				Perm4				Perm3				Perm2				Perm1				Perm0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents Stage 1 Base Permissions.

Perm<m>	Meaning
0b0000	No access, Overlay applied.
0b0001	Read, Overlay applied.
0b0010	Execute, Overlay applied.
0b0011	Read and Execute, Overlay applied.
0b0100	Reserved - treated as No access, Overlay applied.
0b0101	Read and Write, Overlay applied.
0b0110	Read, Write and Execute, Overlay applied.
0b0111	Read, Write and Execute, Overlay applied.
0b1000	Read, Overlay not applied.
0b1001	Read, GCS Read and GCS Write, Overlay not applied.
0b1010	Read and Execute, Overlay not applied.
0b1011	Reserved - treated as No access, Overlay not applied.
0b1100	Read and Write, Overlay not applied.
0b1101	Reserved - treated as No access, Overlay not applied.
0b1110	Read, Write and Execute, Overlay not applied.
0b1111	Reserved - treated as No access, Overlay not applied.

This field is permitted to be cached in a TLB.

When Stage 1 Indirect Permission mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PIR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PIR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIR_EL3;

```

MSR PIR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    PIR_EL3 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PIRE0_EL1, Permission Indirection Register 0 (EL1)

The PIRE0_EL1 characteristics are:

Purpose

Stage 1 Permission Indirection Register for unprivileged access of the EL1&0 translation regime.

Configuration

This register is present only when FEAT_S1PIE is implemented. Otherwise, direct accesses to PIRE0_EL1 are UNDEFINED.

Attributes

PIRE0_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8																								
Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents Stage 1 Base Permissions.

Perm<m>	Meaning
0b0000	No access, Overlay applied.
0b0001	Read, Overlay applied.
0b0010	Execute, Overlay applied.
0b0011	Read and Execute, Overlay applied.
0b0100	Reserved - treated as No access, Overlay applied.
0b0101	Read and Write, Overlay applied.
0b0110	Read, Write and Execute, Overlay applied.
0b0111	Read, Write and Execute, Overlay applied.
0b1000	Read, Overlay not applied.
0b1001	Read, GCS Read and GCS Write, Overlay not applied.
0b1010	Read and Execute, Overlay not applied.
0b1011	Reserved - treated as No access, Overlay not applied.
0b1100	Read and Write, Overlay not applied.
0b1101	Reserved - treated as No access, Overlay not applied.
0b1110	Read, Write and Execute, Overlay not applied.
0b1111	Reserved - treated as No access, Overlay not applied.

This field is permitted to be cached in a TLB.

When Stage 1 Indirect Permission mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PIRE0_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PIRE0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPIRE0_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x290];
    else
        X[t, 64] = PIRE0_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PIRE0_EL2;
    else
        X[t, 64] = PIRE0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIRE0_EL1;

```

MSR PIRE0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPIRE0_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x290] = X[t, 64];
    else
        PIRE0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        PIRE0_EL2 = X[t, 64];
    else
        PIRE0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIRE0_EL1 = X[t, 64];

```

MRS <Xt>, PIRE0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x290];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PIRE0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = PIRE0_EL1;
    else
        UNDEFINED;

```

MSR PIRE0_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x290] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PIRE0_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        PIRE0_EL1 = X[t, 64];
    else
        UNDEFINED;

```


no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PIRE0_EL2, Permission Indirection Register 0 (EL2)

The PIRE0_EL2 characteristics are:

Purpose

Stage 1 Permission Indirection Register for unprivileged access of the EL2&0 translation regime.

Configuration

This register is present only when FEAT_S1PIE is implemented. Otherwise, direct accesses to PIRE0_EL2 are UNDEFINED.

Attributes

PIRE0_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15				Perm14				Perm13				Perm12				Perm11				Perm10				Perm9				Perm8			
Perm7				Perm6				Perm5				Perm4				Perm3				Perm2				Perm1				Perm0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents Stage 1 Base Permissions.

Perm<m>	Meaning
0b0000	No access, Overlay applied.
0b0001	Read, Overlay applied.
0b0010	Execute, Overlay applied.
0b0011	Read and Execute, Overlay applied.
0b0100	Reserved - treated as No access, Overlay applied.
0b0101	Read and Write, Overlay applied.
0b0110	Read, Write and Execute, Overlay applied.
0b0111	Read, Write and Execute, Overlay applied.
0b1000	Read, Overlay not applied.
0b1001	Read, GCS Read and GCS Write, Overlay not applied.
0b1010	Read and Execute, Overlay not applied.
0b1011	Reserved - treated as No access, Overlay not applied.
0b1100	Read and Write, Overlay not applied.
0b1101	Reserved - treated as No access, Overlay not applied.
0b1110	Read, Write and Execute, Overlay not applied.
0b1111	Reserved - treated as No access, Overlay not applied.

This field is permitted to be cached in a TLB.

When Stage 1 Indirect Permission mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PIRE0_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name PIRE0_EL2 or [PIRE0_EL1](#) are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PIRE0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x298];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PIRE0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIRE0_EL2;

```

MSR PIRE0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x298] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PIRE0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIRE0_EL2 = X[t, 64];

```

MRS <Xt>, PIRE0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPIRE0_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x290];
    else
        X[t, 64] = PIRE0_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PIRE0_EL2;
    else
        X[t, 64] = PIRE0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PIRE0_EL1;

```

MSR PIRE0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPIRE0_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x290] = X[t, 64];
    else
        PIRE0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        PIRE0_EL2 = X[t, 64];
    else
        PIRE0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PIRE0_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PM, PMU Exception Mask

The PM characteristics are:

Purpose

Allows access to the PMU exception Mask bit.

Configuration

This register is present only when FEAT_SEBEP is implemented. Otherwise, direct accesses to PM are UNDEFINED.

Attributes

PM is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																PM																
RES0																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:33]

Reserved, RES0.

PM, bit [32]

PMU Exception Mask.

PM	Meaning
0b0	Does not cause the PMU exception to be masked.
0b1	Causes the PMU exception to be masked.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [31:0]

Reserved, RES0.

Accessing PM

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PM

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(31):PSTATE.PM:Zeros(32);
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(31):PSTATE.PM:Zeros(32);
elsif PSTATE.EL == EL3 then
    X[t, 64] = Zeros(31):PSTATE.PM:Zeros(32);

```

MSR PM, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PSTATE.PM = X[t, 64]<32>;
elsif PSTATE.EL == EL2 then
    PSTATE.PM = X[t, 64]<32>;
elsif PSTATE.EL == EL3 then
    PSTATE.PM = X[t, 64]<32>;

```

MSR PM, #<imm>

op0	op1	CRn	CRm	op2
0b00	0b001	0b0100	0b001x	0b000

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMBIDR_EL1, Profiling Buffer ID Register

The PMBIDR_EL1 characteristics are:

Purpose

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBIDR_EL1 are UNDEFINED.

Attributes

PMBIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
RES0												EA				RES0		F	P	Align											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:12]

Reserved, RES0.

EA, bits [11:8]

External Abort handling. Describes how the PE manages External aborts on writes made by the Statistical Profiling Unit to the Profiling Buffer.

EA	Meaning
0b0000	Not described.
0b0001	The PE ignores External aborts on writes made by the Statistical Profiling Unit.
0b0010	The External abort generates an SError interrupt at the PE.

All other values are reserved.

From Armv8.8, the value 0b0000 is not permitted.

PMBIDR_EL1.EA describes only External aborts generated by the write to memory. External aborts on a translation table walk made by the Statistical Profiling Unit generate Profiling Buffer management events reported as MMU faults using [PMBSR_EL1](#).

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [7:6]

Reserved, RES0.

F, bit [5]

Flag updates. Describes how address translations performed by the Statistical Profiling Unit manage the Access flag and dirty state.

F	Meaning
0b0	Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Unit is always disabled for all translation stages.
0b1	Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Unit is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.

Note

If hardware management of the Access flag is disabled for a stage of translation, an access to a Page or Block with the Access flag bit not set in the descriptor will generate an Access Flag fault.

If hardware management of the dirty state is disabled for a stage of translation, an access to a Page or Block will ignore the Dirty Bit Modifier in the descriptor and might generate a Permission fault, depending on the values of the access permission bits in the descriptor.

From Armv8.8, the value 0 is not permitted.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

P, bit [4]

Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the Profiling Buffer is owned by a higher Exception level or another Security state. Defined values are:

P	Meaning
0b0	Programming is allowed.
0b1	Programming not allowed.

The value read from this field depends on the current Exception level and the Effective values of [MDCR_EL3.NSPB](#), [MDCR_EL3.NSPBE](#), and [MDCR_EL2.E2PB](#):

- If EL3 is implemented, and the owning Security state is Secure state, this field reads as one from:
 - Non-secure EL1 and Non-secure EL2.
 - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
 - If Secure EL2 is implemented and enabled, and [MDCR_EL2.E2PB](#) is 0b00, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this field reads as one from:
 - Secure EL1.
 - If Secure EL2 is implemented, Secure EL2.
 - If EL2 is implemented and [MDCR_EL2.E2PB](#) is 0b00, Non-secure EL1.
 - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this field reads as one from:
 - Non-secure EL1 and Non-secure EL2.
 - Secure EL1 and Secure EL2.
 - If [MDCR_EL2.E2PB](#) is 0b00, Realm EL1.
- If EL3 is not implemented, EL2 is implemented, and [MDCR_EL2.E2PB](#) is 0b00, this field reads as one from EL1.
- Otherwise, this field reads as zero.

Align, bits [3:0]

Defines the minimum alignment constraint for writes to [PMBPTR_EL1](#). Defined values are:

Align	Meaning
0b0000	Byte.
0b0001	Halfword.
0b0010	Word.
0b0011	Doubleword.
0b0100	16 bytes.
0b0101	32 bytes.
0b0110	64 bytes.
0b0111	128 bytes.
0b1000	256 bytes.
0b1001	512 bytes.
0b1010	1KB.
0b1011	2KB.

All other values are reserved.

For more information, see 'Restrictions on the current write pointer'.

If this field is non-zero, then every record is a multiple of this size.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMBIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMBIDR EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBIDR_EL1;

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMBLIMITR_EL1, Profiling Buffer Limit Address Register

The PMBLIMITR_EL1 characteristics are:

Purpose

Defines the upper limit for the profiling buffer, and enables the profiling buffer

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBLIMITR_EL1 are UNDEFINED.

Attributes

PMBLIMITR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																LIMIT																					
LIMIT																RES0						PMFZ	RES0	FM	E												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

LIMIT, bits [63:12]

Limit address. PMBLIMITR_EL1.LIMIT:Zeros(12) is the address of the first byte in memory after the last byte in the profiling buffer. If the smallest implemented translation granule is not 4KB, then bits[N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value, $\text{Log}_2(\text{smallest implemented translation granule})$.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:6]

Reserved, RES0.

PMFZ, bit [5]

When FEAT_SPEv1p2 is implemented:

Freeze PMU on SPE event. Stop PMU event counters when [PMBSR_EL1.S](#) == 1.

PMFZ	Meaning
0b0	Do not freeze PMU event counters on Statistical Profiling Buffer Management event.
0b1	Freeze PMU event counters on Statistical Profiling Buffer Management event.

The PMU event counters affected by this control is controlled by [PMCR_ELO.FZS](#) and, if EL2 is implemented, [MDCR_EL2.HPMFZS](#). See the descriptions of these control bits for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [4:3]

Reserved, RES0.

FM, bits [2:1]

Fill mode.

FM	Meaning	Applies when
0b00	Fill mode. Stop collection and raise maintenance interrupt on buffer fill.	
0b10	Discard mode. All output is discarded.	When FEAT_SPEv1p2 is implemented

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Profiling Buffer enable

E	Meaning
0b0	All output is discarded.
0b1	Profiling buffer enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMBLIMITR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMBLIMITR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMBLIMITR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x800];
    else
        X[t, 64] = PMBLIMITR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMBLIMITR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBLIMITR_EL1;

```

MSR PMBLIMITR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMBLIMITR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x800] = X[t, 64];
    else
        PMBLIMITR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMBLIMITR_EL1 = X[t, 64];
elseif PSTATE.EL == EL3 then
    PMBLIMITR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMBPTR_EL1, Profiling Buffer Write Pointer Register

The PMBPTR_EL1 characteristics are:

Purpose

Defines the current write pointer for the profiling buffer.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBPTR_EL1 are UNDEFINED.

Attributes

PMBPTR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																PTR															
																PTR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PTR, bits [63:0]

Current write address. Defines the virtual address of the next entry to be written to the buffer.

The architecture places restrictions on the values software can write to the pointer. For more information see 'Restrictions on the current write pointer'.

Note

As a result, an implementation might treat some of bits[M:0], where M is defined by [PMBIDR_EL1](#).Align, as RES0.

On a management interrupt, PMBPTR_EL1 is frozen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMBPTR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMBPTR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMBPTR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x810];
    else
        X[t, 64] = PMBPTR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMBPTR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBPTR_EL1;

```

MSR PMBPTR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMBPTR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
            NVMem[0x810] = X[t, 64];
        else
            PMBPTR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMBPTR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMBPTR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMBSR_EL1, Profiling Buffer Status/syndrome Register

The PMBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software when the buffer is disabled because the management interrupt has been raised.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMBSR_EL1 are UNDEFINED.

Attributes

PMBSR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																								AssuredOnly		Overlay		DirtyBit				RES0	
EC				RES0				DL		EA		S		COLL		MSS																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:40]32

Reserved, RES0.

AssuredOnly, bit [39]

When FEAT_THE is implemented:

AssuredOnly flag.

If a memory access generates a Stage 2 Data Abort, this field holds information about the fault.

AssuredOnly	Meaning
0b0	The Data Abort is not due to AssuredOnly.
0b1	The Data Abort is due to AssuredOnly.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Overlay, bit [38]**When FEAT_S1POE is implemented or FEAT_S2POE is implemented:**

Overlay flag.

If a memory access generates a Data Abort for a Permission fault, this field holds information about the fault.

Overlay	Meaning
0b0	The Data Abort is due to Base Permissions.
0b1	The Data Abort is due to Overlay Permissions.

For any other fault, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DirtyBit, bit [37]**When FEAT_S1PIE is implemented or FEAT_S2PIE is implemented:**

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, this field holds information about the fault.

DirtyBit	Meaning
0b0	The Permission Fault is not due to nDirty State or Dirty State.
0b1	The Permission Fault is due to nDirty State or Dirty State.

For any other fault or Access, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [36:32]

Reserved, RES0.

EC, bits [31:26]

Event class. Top-level description of the cause of the buffer management event.

EC	Meaning	MSS	Applies when
0b000000	Other buffer management event. All buffer management events other than those described by other defined Event class codes.	MSS encoding for other buffer management events	
0b011110	Granule Protection Check fault, other than GPF, on write to Profiling Buffer.	MSS encoding for Granule Protection Check fault	When FEAT_RME is implemented
0b011111	Buffer management event for an IMPLEMENTATION DEFINED reason.	MSS encoding for a buffer management event for an IMPLEMENTATION DEFINED reason	
0b100100	Stage 1 Data Abort on write to Profiling Buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to buffer	
0b100101	Stage 2 Data Abort on write to Profiling Buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to buffer	

All other values are reserved. Reserved values might be defined in a future version of the architecture.

Writing a reserved value to this field will make the value of this field UNKNOWN. Values that are not supported act as reserved values when writing to this register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [25:20]

Reserved, RES0.

DL, bit [19]

Partial record lost.

Following a buffer management event other than an asynchronous External abort, indicates whether the last record written to the Profiling Buffer is complete.

DL	Meaning
0b0	PMBPTR_EL1 points to the first byte after the last complete record written to the Profiling Buffer.
0b1	Part of a record was lost because of a buffer management event or synchronous External abort. PMBPTR_EL1 might not point to the first byte after the last complete record written to the buffer, and so restarting collection might result in a data record stream that software cannot parse. All records prior to the last record have been written to the buffer.

When the buffer management event was because of an asynchronous External abort, this bit is set to 1 and software must not assume that any valid data has been written to the Profiling Buffer.

This bit is RES0 if the PE never sets this bit as a result of a buffer management event caused by an asynchronous External abort.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EA, bit [18]

External abort.

EA	Meaning
0b0	An External abort has not been asserted.
0b1	An External abort has been asserted and detected by the Statistical Profiling Unit.

This bit is RES0 if the PE never sets this bit as the result of an External abort.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

S, bit [17]

Service

S	Meaning
0b0	PMBIRQ is not asserted.
0b1	PMBIRQ is asserted. All profiling data has either been written to the buffer or discarded.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

COLL, bit [16]

Collision detected.

COLL	Meaning
0b0	No collision events detected.
0b1	At least one collision event was recorded.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

MSS, bits [15:0]

Management Event Specific Syndrome.

Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

MSS encoding for stage 1 or stage 2 Data Aborts on write to buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										FSC					

Bits [15:6]

Reserved, RES0.

FSC, bits [5:0]

Fault status code

FSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Asynchronous External abort.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented

0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

It is IMPLEMENTATION DEFINED whether each of the Access Flag fault, asynchronous External abort and synchronous External abort, Alignment fault, and TLB Conflict abort values can be generated by the PE. For more information see 'Faults and Watchpoints'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

MSS encoding for other buffer management events

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										BSC					

Bits [15:6]

Reserved, RES0.

BSC, bits [5:0]

Buffer status code

BSC	Meaning
0b000000	Buffer not filled
0b000001	Buffer filled

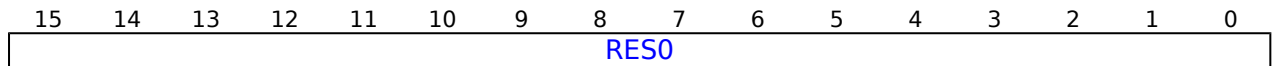
All other values are reserved. Reserved values might be defined in a future version of the architecture.

Writing a reserved value to this field will make the value of this field UNKNOWN. Values that are not supported act as reserved values when writing to this register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

MSS encoding for Granule Protection Check fault



Bits [15:0]

Reserved, RES0.

MSS encoding for a buffer management event for an IMPLEMENTATION DEFINED reason



IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMBSR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMBSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x820];
    else
        X[t, 64] = PMBSR_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMBSR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = PMBSR_EL1;

```

MSR PMBSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2PB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
            NVMem[0x820] = X[t, 64];
        else
            PMBSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMBSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMBSR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCCFILTR_EL0, Performance Monitors Cycle Count Filter Register

The PMCCFILTR_EL0 characteristics are:

Purpose

Determines the modes in which the Cycle Counter, [PMCCNTR_EL0](#), increments.

Configuration

AArch64 System register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCCFILTR\[31:0\]](#).

AArch64 System register PMCCFILTR_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMCCFILTR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

AArch64 System register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMCCFILTR_EL0\[31:0\]](#)[PMCCFILTR_EL0\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCCFILTR_EL0 are UNDEFINED.

Attributes

PMCCFILTR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
P	U	NSK	NSU	NSH	M	RES0	SH	T	RLK	RLU	RLH	RES0																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

P	Meaning
0b0	Count cycles in EL1.
0b1	Do not count cycles in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

U	Meaning
0b0	Count cycles in EL0.
0b1	Do not count cycles in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

NSH	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

M, bit [26]

When EL3 is implemented:

Secure EL3 filtering bit.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [25]

Reserved, RES0.

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted.

Otherwise, cycles in Secure EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

T, bit [23]

When FEAT_TME is implemented:

Transactional state filtering bit. Controls counting of Attributable events in Non-transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.cycles.
0b1	Do not count Attributable eventscycles in Non-transactional state.

For each Unattributable event, it is IMPLEMENTATION DEFINED whether the filtering applies.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted.

Otherwise, cycles in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted.

Otherwise, cycles in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:0]

Reserved, RES0.

Accessing PMCCFILTR_EL0

PMCCFILTR_EL0 can also be accessed by using [PMXEVTYPERS_EL0](#) with [PMSELR_EL0](#).SEL set to 0b11111.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCCFILTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCFILTR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCFILTR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCFILTR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCCFILTR_EL0;

```

MSR PMCCFILTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCCFILTR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCCNTR_EL0, Performance Monitors Cycle Count Register

The PMCCNTR_EL0 characteristics are:

Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' for more information.

[PMCCFILTR_EL0](#) determines the modes and states in which the PMCCNTR_EL0 can increment.

Configuration

AArch64 System register PMCCNTR_EL0 bits [63:0] are architecturally mapped to AArch32 System register [PMCCNTR\[63:0\]](#).

AArch64 System register PMCCNTR_EL0 bits [63:0] are architecturally mapped to External register [PMU.PMCCNTR_EL0\[63:0\]](#)[PMCCNTR_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCCNTR_EL0 are UNDEFINED.

All counters are subject to any changes in clock frequency, including clock stopping caused by the WFI and WFE instructions. This means that it is CONSTRAINED UNPREDICTABLE whether or not PMCCNTR_EL0 continues to increment when clocks are stopped by WFI and WFE instructions.

Attributes

PMCCNTR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CCNT, bits [63:0]

Cycle count. Depending on the values of [PMCR_EL0](#).{LC,D}, this field increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR_EL0](#).C sets this field to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCCNTR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCCNTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<CR,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCCNTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCNTR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCCNTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCNTR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCCNTR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCCNTR_EL0;

```

MSR PMCCNTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCCNTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCNTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMCCNTR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCNTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCCNTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCCNTR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMCCNTSVR_EL1, Performance Monitors Cycle Count Saved Value Register

The PMCCNTSVR_EL1 characteristics are:

Purpose

Captures the PMU Cycle counter, [PMCCNTR_EL0](#).

Configuration

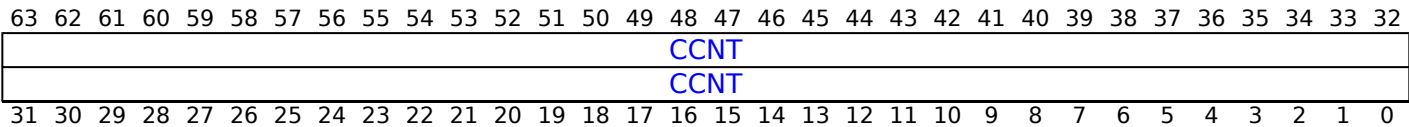
AArch64 System register PMCCNTSVR_EL1 bits [63:0] are architecturally mapped to External register [PMU.PMCCNTSVR_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMCCNTSVR_EL1 are UNDEFINED.

Attributes

PMCCNTSVR_EL1 is a 64-bit register.

Field descriptions



CCNT, bits [63:0]

Sampled Cycle Count. The value of [PMCCNTR_EL0](#) at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCCNTSVR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCCNTSVR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1110	0b1011	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMCCNTSVR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMCCNTSVR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCCNTSVR_EL1;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

PMCEID0_EL0, Performance Monitors Common Event Identification register 0

The PMCEID0_EL0 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see 'The PMU event number space and common events'.

Configuration

AArch64 System register PMCEID0_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID0\[31:0\]](#).

AArch64 System register PMCEID0_EL0 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID2\[31:0\]](#).

AArch64 System register PMCEID0_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMCEID0\[31:0\]](#)[PMCEID0\[31:0\]](#).

AArch64 System register PMCEID0_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMCEID2\[31:0\]](#)[PMCEID2\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCEID0_EL0 are UNDEFINED.

Attributes

PMCEID0_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47
IDhi31	IDhi30	IDhi29	IDhi28	IDhi27	IDhi26	IDhi25	IDhi24	IDhi23	IDhi22	IDhi21	IDhi20	IDhi19	IDhi18	IDhi17	IDhi16	IDhi15
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15

IDhi<n>, bit [n+32], for n = 31 to 0

When FEAT_PMUv3p1 is implemented:

IDhi[n] corresponds to Common event (0x4000 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Otherwise:

Reserved, RES0.

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event n.

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Accessing PMCEID0_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCEID0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b110

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID0_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID0_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID0_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCEID0_EL0;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID1_EL0, Performance Monitors Common Event Identification register 1

The PMCEID1_EL0 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see 'The PMU event number space and common events'.

Configuration

AArch64 System register PMCEID1_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID1\[31:0\]](#).

AArch64 System register PMCEID1_EL0 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID3\[31:0\]](#).

AArch64 System register PMCEID1_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMCEID1\[31:0\]](#)[PMCEID1\[31:0\]](#).

AArch64 System register PMCEID1_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMCEID3\[31:0\]](#)[PMCEID3\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCEID1_EL0 are UNDEFINED.

Attributes

PMCEID1_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47
IDhi31	IDhi30	IDhi29	IDhi28	IDhi27	IDhi26	IDhi25	IDhi24	IDhi23	IDhi22	IDhi21	IDhi20	IDhi19	IDhi18	IDhi17	IDhi16	IDhi15
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15

IDhi<n>, bit [n+32], for n = 31 to 0

When FEAT_PMUv3p1 is implemented:

IDhi[n] corresponds to Common event (0x4020 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Otherwise:

Reserved, RES0.

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Accessing PMCEID1_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCEID1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID1_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID1_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCEID1_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCEID1_EL0;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCNTENCLR_EL0, Performance Monitors Count Enable Clear register

The PMCNTENCLR_EL0 characteristics are:

Purpose

Disables the Cycle Count Register, [PMCCNTR_EL0](#), and any implemented event counters [PMEVCNTR<n>_EL0](#). Reading this register shows which counters are enabled.

Configuration

AArch64 System register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENCLR\[31:0\]](#).

AArch64 System register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMCNTENCLR_EL0\[31:0\]](#) [PMCNTENCLR_EL0\[31:0\]](#).

AArch64 System register PMCNTENCLR_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMCNTENCLR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCNTENCLR_EL0 are UNDEFINED.

Attributes

PMCNTENCLR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:[33](#)[32](#)]

Reserved, RES0.

F<m>, bit [m+32], for m = 0

When FEAT_PMUv3_ICNTR is implemented:

Fixed-function counter <m> disable. On writes, allows software to disable fixed-function counter <m>. On reads, returns the fixed-function counter <m> enable status.

F<m>	Meaning
0b0	Fixed-function counter <m> disabled.
0b1	Fixed-function counter <m> enabled.

PMCNTENCLR_EL0.F0 holds the enable status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

[PMCCNTR_EL0](#) disable bit. Disables the cycle counter register. Possible values are:

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, disables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter disable bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that PMEVCNTR<n>_EL0 is enabled. When written, disables PMEVCNTR<n>_EL0 .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENCLR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCNTENCLR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b010

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENCLR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENCLR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENCLR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCNTENCLR_EL0;

```

MSR PMCNTENCLR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b010


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENCLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENCLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENCLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCNTENCLR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCNTENSET_EL0, Performance Monitors Count Enable Set register

The PMCNTENSET_EL0 characteristics are:

Purpose

Enables the Cycle Count Register, [PMCCNTR_EL0](#), and any implemented event counters [PMEVCNTR<n>_EL0](#). Reading this register shows which counters are enabled.

Configuration

AArch64 System register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENSET\[31:0\]](#).

AArch64 System register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMCNTENSET_EL0\[31:0\]](#) [PMCNTENSET_EL0\[31:0\]](#).

AArch64 System register PMCNTENSET_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMCNTENSET_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCNTENSET_EL0 are UNDEFINED.

Attributes

PMCNTENSET_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:**33**32]

Reserved, RES0.

F<m>, bit [m+32], for m = 0

When FEAT_PMUv3_ICNTR is implemented:

Fixed-function counter <m> enable. On writes, allows software to enable fixed-function counter <m>. On reads, returns the fixed-function counter <m> enable status.

F<m>	Meaning
0b0	Fixed-function counter <m> disabled.
0b1	Fixed-function counter <m> enabled.

PMCNTENSET_EL0.F0 holds the enable status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **WIS**.

Otherwise:

Reserved, RES0.

C, bit [31]

[PMCCNTR_EL0](#) enable bit. Enables the cycle counter register. Possible values are:

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, enables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter enable bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that PMEVCNTR<n>_EL0 event counter is enabled. When written, enables PMEVCNTR<n>_EL0 .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENSET_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCNTENSET_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENSET_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENSET_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCNTENSET_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCNTENSET_EL0;

```

MSR PMCNTENSET_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMCNTEN == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCNTENSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCNTENSET_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCR_EL0, Performance Monitors Control Register

The PMCR_EL0 characteristics are:

Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configuration

AArch64 System register PMCR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCR\[31:0\]](#).

AArch64 System register PMCR_EL0 bits [7:0] are architecturally mapped to External register [PMU.PMCR_EL0\[7:0\]](#)~~PMCR_EL0[7:0]~~.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCR_EL0 are UNDEFINED.

Attributes

PMCR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															FZS
IMP								IDCODE								N				RES0	FZ0	RES0	LP	LC	DP	X	D	C	P	E	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

FZS, bit [32]

When FEAT_SPEv1p2 is implemented:

Freeze-on-SPE event. Stop counters when [PMBLIMITR_EL1](#).{PMFZ,E} == {1,1} and [PMBSR_EL1](#).S == 1.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

FZS	Meaning
0b0	Do not freeze on Statistical Profiling Buffer Management event.
0b1	Event counter PMEVCNTR<n>_EL0 does not count following a Statistical Profiling Buffer Management event if n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - When the implementation only supports execution in AArch64 state, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

IMP, bits [31:24]

When FEAT_PMUv3p7 is not implemented:

Implementer code.

If this field is zero, then PMCR_EL0.IDCODE is RES0 and software must use [MIDR_EL1](#) to identify the PE.

Otherwise, this field and PMCR_EL0.IDCODE identify the PMU implementation to software. The implementer codes are allocated by Arm. A non-zero value has the same interpretation as [MIDR_EL1](#).Implementer.

Use of this field is deprecated.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RAZ.

IDCODE, bits [23:16]

When PMCR_EL0.IMP != 0b00000000:

Identification code. Use of this field is deprecated.

Each implementer must maintain a list of identification codes that are specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

N, bits [15:11]

Indicates the number of event counters implemented. This value is in the range of 0b000000-0b11111. If the value is 0b000000, then only [PMCCNTR_EL0](#) is implemented. If the value is 0b11111, then [PMCCNTR_EL0](#) and 31 event counters are implemented.

When EL2 is implemented and enabled for the current Security state, reads of this field from EL1 and EL0 return the value of [MDCR_EL2](#).HPMN.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bit [10]

Reserved, RES0.

FZO, bit [9]

When FEAT_PMUv3p7 is implemented:

Freeze-on-overflow. Stop event counters on overflow.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counter PMEVCNTR<n>_EL0 does not count when PMOVSCLR_EL0 [(PMN-1):0] is nonzero and n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [8]

Reserved, RES0.

LP, bit [7]

When FEAT_PMUv3p5 is implemented:

Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

LP	Meaning
0b0	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0 [31:0].
0b1	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n>_EL0 [63:0].

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LC, bit [6]

When AArch32 is supported:

Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit.

LC	Meaning
0b0	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR_EL0 [31:0].
0b1	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR_EL0 [63:0].

Arm deprecates use of [PMCR_EL0](#).LC = 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

DP, bit [5]

When EL3 is implemented or (FEAT_PMUv3p1 is implemented and EL2 is implemented):

Disable cycle counter when event counting is prohibited.

DP	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL2. If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL3. If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMCCNTR_EL0 is disabled. If EL3 is implemented, MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or MDCR_EL3.MPMX is 0, then cycle counting by PMCCNTR_EL0 is disabled at EL3 and in Secure state. <p>If MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(MDCR_EL2.HPMN-1)] is prohibited or frozen.</p>

For more information, see 'Prohibiting event and cycle counting'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

X, bit [4]

When the implementation includes a PMU event export bus:

Enable export of events in an IMPLEMENTATION DEFINED PMU event export bus.

X	Meaning
0b0	Do not export events.
0b1	Export events where not prohibited.

This field enables the exporting of events over an IMPLEMENTATION DEFINED PMU event export bus to another device, for example to an OPTIONAL trace unit.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

D, bit [3]

When AArch32 is supported:

Clock divider.

D	Meaning
0b0	When enabled, PMCCNTR_EL0 counts every clock cycle.
0b1	When enabled, PMCCNTR_EL0 counts once every 64 clock cycles.

If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

Arm deprecates use of PMCR_EL0.D = 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

C, bit [2]

Cycle counter reset. The effects of writing to this bit are:

C	Meaning
0b0	No action.
0b1	Reset PMCCNTR_EL0 to zero.

Note

Resetting [PMCCNTR_EL0](#) does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset.

Access to this field is **WO/RAZ**.

P, bit [1]

Event counter reset.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

P	Meaning
0b0	No action.
0b1	If n is in the range of affected event counters, resets each event counter PMEVCNTR<n>_EL0 to zero.

The effects of writing to this bit are:

- If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)].
- If EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters.
- In EL2 and EL3, a write of 1 to this bit resets all the event counters.
- This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

Note

Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the values of [MDCR_EL2](#).HLP and PMCR_EL0.LP are ignored, and bits [63:0] of all affected event counters are reset.

Access to this field is **WO/RAZ**.

E, bit [0]

Enable.

If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.

If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.

If EL2 is not implemented, PMN is PMCR_EL0.N.

E	Meaning
0b0	PMCCNTR_EL0 is disabled and event counters PMEVCNTR<n>_EL0 , where n is in the range of affected event counters, are disabled.
0b1	PMCCNTR_EL0 and event counters PMEVCNTR<n>_EL0 , where n is in the range of affected event counters, are enabled by PMCNTENSET_EL0 .

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMCR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMCR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMCR_EL0;

```

MSR PMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMCR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMECR_EL1, Performance Monitors Exception Control Register

The PMECR_EL1 characteristics are:

Purpose

Controls the behavior on overflow of the performance monitors.

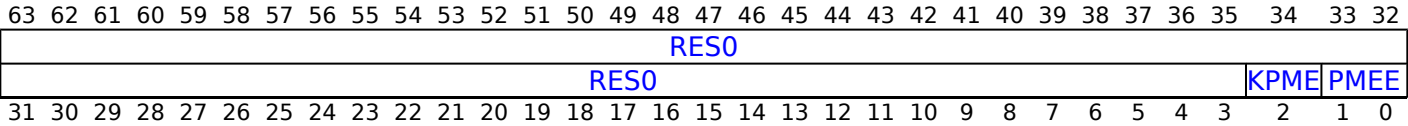
Configuration

This register is present only when FEAT_PMuV3 is implemented and FEAT_EBEP is implemented. Otherwise, direct accesses to PMECR_EL1 are UNDEFINED.

Attributes

PMECR_EL1 is a 64-bit register.

Field descriptions



Bits [63:3]

Reserved, RES0.

KPME, bit [2]

Local (Kernel) PMU Exception Enable. Enables PMU exceptions taken to the current Exception level.

KPME	Meaning
0b0	PMU exceptions taken to the current Exception level are disabled.
0b1	PMU exceptions taken to the current Exception level are not affected by this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PMEE, bits [1:0]

Performance Monitors Exception Enable. Controls the generation of **PMUIRQ** signal and PMU exception at EL0 and EL1.

PMEE	Meaning
0b00	PMUIRQ signal is enabled, and PMU exception is disabled.
0b10	PMUIRQ signal is disabled, and PMU exception is disabled.
0b11	PMUIRQ signal is disabled, and PMU exception is enabled.

All other values are reserved.

This field is ignored by the PE when any of the following are true:

- All of the following are true:
 - EL3 is implemented.
 - [MDCR_EL3](#).PMEE != 0b01.
- All of the following are true:
 - EL2 is implemented and enabled in the current Security State.
 - [MDCR_EL2](#).PMEE != 0b01.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMECR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMECR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMECR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMECR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMECR_EL1;
```

MSR PMECR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PMECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMECR_EL1 = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVCNTR<n>_EL0

PMEVCNTR<n>_EL0 can also be accessed by using [PMXEVCNTR_EL0](#) with [PMSELR_EL0](#).SEL set to the value of <n>.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMEVCNTR<n>_EL0](#) is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of [PMEVCNTR<n>_EL0](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR_EL0](#).{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, [MDCR_EL2](#).HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [MDCR_EL2](#).HPMN.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMEVCNTR<m>_EL0 ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b10:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMEVCNTR_EL0[m];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[m];

```

MSR PMEVCNTR<m>_EL0, <Xt> ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b10:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elseif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVCNTR_EL0[m] = X[t, 64];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[m] = X[t, 64];
elseif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[m] = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

PMEVCNTSVR<n>_EL1, Performance Monitors Event Count Saved Value Register <n>, n = 0 - 30

The PMEVCNTSVR<n>_EL1 characteristics are:

Purpose

Captures the PMU Event counter <n>, [PMEVCNTR<n>_EL0](#).

Configuration

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMEVCNTSVR<n>_EL1 are UNDEFINED.

Attributes

PMEVCNTSVR<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																EVCNT															
																EVCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EVCNT, bits [63:0]

Sampled Event Count. The value of [PMEVCNTR<n>_EL0](#) at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVCNTSVR<n>_EL1

If <n> is greater-than-or-equal-to the number of implemented event counters, then direct reads of PMEVCNTSVR<n>_EL1 are UNDEFINED.

Otherwise, direct reads of PMEVCNTSVR<n>_EL1 generate a Trap exception to EL2 when all of the following are true:

- <n> is greater-than-or-equal-to the number of snapshot registers accessible at the current Exception level.
- EL2 is implemented and enabled in the current Security state.
- The access is from EL1.

Note

If EL2 is implemented and enabled in the current Security state, [MDCR_EL2](#).HPMN identifies the number of accessible snapshot registers at EL1. Otherwise, the number of accessible snapshot registers is the number of implemented event counters. See [MDCR_EL2](#).HPMN for more details.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMEVCNTSVR<m>_EL1 ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b10	0b000	0b1110	0b10:m[4:3]	m[2:0]

```
integer m = UInt(CRm<1:0>:op2<2:0>);
```

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMEVCNTSVR_EL1[m];
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMEVCNTSVR_EL1[m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTSVR_EL1[m];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMEVTYPER<n>_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>_EL0 characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Configuration

AArch64 System register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMEVTYPER<n>\[31:0\]](#).

AArch64 System register PMEVTYPER<n>_EL0 bits [63:0] are architecturally mapped to External register [PMU.PMEVTYPER<n>_EL0\[63:0\]](#)[PMEVTYPER<n>_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVTYPER<n>_EL0 are UNDEFINED.

Attributes

PMEVTYPER<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60		59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
TC			TERES0		RES0		TH		SYNC		RES0												TH										
P	U	NSK	NSU		NSH		M	MT	SH	T	RLK	RLU	RLH	RES0			evtCount[15:10]						evtCount[9:0]										
31	30	29	28		27		26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TC, bits [63:61]

When (FEAT_PMUv3_EDGE is not implemented or PMEVTYPER<n>_EL0.TE == 0) and FEAT_PMUv3_TH is implemented:

Threshold Control. Defines the threshold function. In the description of this field, the value V is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled. Comparisons treat V and PMEVTYPER<n>_EL0.TH as unsigned integer values.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b000	Not-equal. The counter increments by V on each processor cycle when V is not equal to PMEVTYPER<n>_EL0.TH. If PMEVTYPER<n>_EL0.TH is zero, the threshold function is disabled. B on each processor cycle when V _B is not equal to TH. If TH is zero, the threshold function is disabled.
0b001	Not-equal, count. The counter increments by 1 on each processor cycle when V is not equal to PMEVTYPER<n>_EL0.TH. B is not equal to TH.
0b010	Equals. The counter increments by V on each processor cycle when V is equal to PMEVTYPER<n>_EL0.TH. B on each processor cycle when V _B is equal to TH.
0b011	Equals, count. The counter increments by 1 on each processor cycle when V is equal to PMEVTYPER<n>_EL0.TH. B is equal to TH.
0b100	Greater-than-or-equal. The counter increments by V on each processor cycle when V is PMEVTYPER<n>_EL0.TH or more. B on each processor cycle when V _B is greater than or equal to TH.
0b101	Greater-than-or-equal, count. The counter increments by 1 on each processor cycle when V is PMEVTYPER<n>_EL0.TH or more. B is greater than or equal to TH.
0b110	Less-than. The counter increments by V on each processor cycle when V is less than PMEVTYPER<n>_EL0.TH. B on each processor cycle when V _B is less than TH.
0b111	Less-than, count. The counter increments by 1 on each processor cycle when V is less than PMEVTYPER<n>_EL0.TH. B is less than TH.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

When FEAT_PMUv3_EDGE is implemented and PMEVTYPER<n>_EL0.TE == 1:

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b001	Equal to not-equal. The counter increments by 1 on each processor cycle when V_B is not equal to TH and V_B was equal to TH on the previous processor cycle.
0b010	Equal to/from not-equal. The counter increments by 1 on each processor cycle when either: <ul style="list-style-type: none"> V_B is not equal to TH and V_B was equal to TH on the previous processor cycle. V_B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b011	Not-equal to equal. The counter increments by 1 on each processor cycle when V_B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b101	Less-than to greater-than-or-equal. The counter increments by 1 on each processor cycle when V_B is greater than or equal to TH and V_B was less than TH on the previous processor cycle.
0b110	Less-than to/from greater-than-or-equal. The counter increments by 1 on each processor cycle when either: <ul style="list-style-type: none"> V_B is greater than or equal to TH and V_B was less than TH on the previous processor cycle. V_B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.
0b111	Greater-than-or-equal to less-than. The counter increments by 1 on each processor cycle when V_B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TE, bit [60]

When FEAT_PMUv3_EDGE is implemented:

Threshold Edge. Enables the edge condition. When PMEVTYPER<n>_EL0.TE is 1, the event counter increments on cycles when the result of the threshold condition changes. See PMEVTYPER<n>_EL0.TC for more information.

TE	Meaning
0b0	Threshold edge condition disabled.
0b1	Threshold edge condition enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BitBits [5960:44]

Reserved, RES0.

SYNC, bit [58]**When FEAT_SEBEP is implemented:**

Synchronous Mode. Controls whether a PMU exception generated by the counter is synchronous or asynchronous.

SYNC	Meaning
0b0	Asynchronous PMU exception is enabled.
0b1	Synchronous PMU exception is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [57:44]

Reserved, RES0.

TH, bits [43:32]**When FEAT_PMUv3_TH is implemented:**

Threshold value. Provides the unsigned value for the threshold function defined by PMEVTYPER<n>_EL0.TC.

If PMEVTYPER<n>_EL0.TC is 0b000 and PMEVTYPER<n>_EL0.TH is zero, then the threshold function is disabled.

If [PMMIR_EL1](#).THWIDTH is less than 12, then bits PMEVTYPER<n>_EL0.TH[11:[PMMIR_EL1](#).THWIDTH] are RES0. This accounts for the behavior when writing a value greater-than-or-equal-to $2^{(\text{PMMIR_EL1.THWIDTH})}$.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMEVTYPER<n>_EL0.RLK bit.

P	Meaning
0b0	Count events in EL1.
0b1	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>_EL0.RLU bit.

U	Meaning
0b0	Count events in EL0.
0b1	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]

When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]

When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMEVTYPER<n>_EL0.RLH bit.

NSH	Meaning
0b0	Do not count events in EL2.
0b1	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

M, bit [26]**When EL3 is implemented:**

EL3 filtering bit.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted.

Otherwise, events in EL3 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MT, bit [25]**When FEAT_MTPMU is implemented or an IMPLEMENTATION DEFINED multi-threaded PMU extension is implemented:**

Multithreading.

MT	Meaning
0b0	Count events only on controlling PE.
0b1	Count events from any PE with the same affinity at level 1 and above as this PE.

From Armv8.6, the IMPLEMENTATION DEFINED multi-threaded PMU extension is not permitted, meaning if FEAT_MTPMU is not implemented, this field is RES0. See [ID_AA64DFR0_EL1](#).MTPMU.

This field is ignored by the PE and treated as zero when FEAT_MTPMU is implemented and Disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SH, bit [24]

When FEAT_SEL2 is implemented and EL3 is implemented:

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted.

Otherwise, events in Secure EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

T, bit [23]

When FEAT_TME is implemented:

Transactional state filtering bit. Controls counting of Attributable events in Non-transactional Transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count Attributable events in Non-transactional Transactional state.

For each Unattributable event, it is IMPLEMENTATION DEFINED whether the filtering applies.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted.

Otherwise, events in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]**When FEAT_RME is implemented:**

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]**When FEAT_RME is implemented:**

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted.

Otherwise, events in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]**When FEAT_PMUv3p1 is implemented:**

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

evtCount[9:0], bits [9:0]

Event to count.

The event number of the event that is counted by event counter [PMEVCNTR<n>_EL0](#).

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

Note

Arm recommends this behavior for all implementations of FEAT_PMUv3.

Otherwise, if PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is UNKNOWN.

Note

UNPREDICTABLE means the event must not expose privileged information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVTYPER<n>_EL0

PMEVTYPER<n>_EL0 can also be accessed by using [PMXEVTYPER_EL0](#) with [PMSELR_EL0](#).SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMEVTYPER<n>_EL0](#) is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of [PMEVTYPER<n>_EL0](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR_EL0](#).EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, [MDCR_EL2](#).HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [MDCR_EL2](#).HPMN.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMEVTYPER<m>_EL0 ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVTYPEPERn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMEVTYPER_EL0[m];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMEVTYPEPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVTYPER_EL0[m];

```

MSR PMEVTYPER<m>_EL0, <Xt> ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVTYPEPERn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVTYPER_EL0[m] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMEVTYPEPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

no old file

htmldiff from-

(new)

PMIAR_EL1, Performance Monitors Instruction Address Register

The PMIAR_EL1 characteristics are:

Purpose

Captures the address of the instruction generating a PMU exception.

Configuration

This register is present only when FEAT_SEBEP is implemented. Otherwise, direct accesses to PMIAR_EL1 are UNDEFINED.

Attributes

PMIAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
EL		RES0							ADDRESS																								
ADDRESS																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

EL, bits [63:62]

Exception Level.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [61:56]

Reserved, RES0.

ADDRESS, bits [55:0]

Instruction virtual address.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMIAR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMIAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMIAR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMIAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMIAR_EL1;

```

MSR PMIAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PMIAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMIAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMIAR_EL1 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMICFILTR_EL0, Performance Monitors Instruction Counter Filter Register

The PMICFILTR_EL0 characteristics are:

Purpose

Configures the Instruction Counter.

Configuration

AArch64 System register PMICFILTR_EL0 bits [63:0] are architecturally mapped to External register [PMU.PMICFILTR_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3 is implemented and FEAT_PMUv3_ICNTR is implemented. Otherwise, direct accesses to PMICFILTR_EL0 are UNDEFINED.

Attributes

PMICFILTR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0					SYNC	RES0																									
P	U	NSK	NSU	NSH	M	RES0	SH	T	RLK	RLU	RLH	RES0				evtCount															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:59]

Reserved, RES0.

SYNC, bit [58]

When FEAT_SEBEP is implemented:

Synchronous mode. Controls whether a PMU exception generated by the counter is synchronous or asynchronous.

SYNC	Meaning
0b0	Asynchronous PMU exception is enabled.
0b1	Synchronous PMU exception is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [57:32]

Reserved, RES0.

P, bit [31]

EL1 filtering. Controls counting instructions in EL1.

P	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in EL1 are not counted.

If EL3 is implemented, then counting instructions in Non-secure EL1 is further controlled by PMICFILTR_EL0.NSK, and counting instructions in EL3 is further controlled by PMICFILTR_EL0.M.

If FEAT_RME is implemented, then counting instructions in Realm EL1 is further controlled by PMICFILTR_EL0.RLK.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

EL0 filtering. Controls counting instructions in EL0.

U	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in EL0 are not counted.

If EL3 is implemented, then PMICFILTR_EL0.NSU further controls filtering instructions in Non-secure EL0.

If FEAT_RME is implemented, then counting instructions in Realm EL1 is further controlled by PMICFILTR_EL0.RLU.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 filtering. Controls counting instructions in Non-secure EL1. If PMICFILTR_EL0.NSK is equal to PMICFILTR_EL0.P, then instructions in Non-secure EL1 are counted. Otherwise, instructions in Non-secure EL1 are not counted.

NSK	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in Non-secure EL1 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in Non-secure EL1 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 filtering. Controls counting instructions in Non-secure EL0. If PMICFILTR_EL0.NSU is equal to PMICFILTR_EL0.U, then instructions in Non-secure EL0 are counted. Otherwise, instructions in Non-secure EL0 are not counted.

NSU	Meaning
0b0	If PMICFILTR_EL0.U == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.U == 1, then instructions in Non-secure EL0 are not counted.
0b1	If PMICFILTR_EL0.U == 0, then instructions in Non-secure EL0 are not counted. If PMICFILTR_EL0.U == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 filtering. Controls counting instructions in EL2.

NSH	Meaning
0b0	Instructions in EL2 are not counted.
0b1	This bit has no effect on filtering of instructions.

If EL3 is implemented and FEAT_SEL2 is implemented, then counting instructions in Secure EL2 is further controlled by PMICFILTR_EL0.SH.

If FEAT_RME is implemented, then counting instructions in Realm EL2 is further controlled by PMICFILTR_EL0.RLH.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

M, bit [26]**When EL3 is implemented:**

EL3 filtering. Controls counting instructions in EL3. If PMICFILTR_EL0.M is equal to PMICFILTR_EL0.P, then instructions in EL3 are counted. Otherwise, instructions in EL3 are not counted.

M	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in EL3 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in EL3 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [25]

Reserved, RES0.

SH, bit [24]

When EL3 is implemented and FEAT_SEL2 is implemented:

Secure EL2 filtering. Controls counting instructions in Secure EL2. If PMICFILTR_EL0.SH is not equal to PMICFILTR_EL0.NSH, then instructions in Secure EL2 are counted. Otherwise, instructions in Secure EL2 are not counted.

SH	Meaning
0b0	If PMICFILTR_EL0.NSH == 0, then instructions in Secure EL2 are not counted. If PMICFILTR_EL0.NSH == 1, then this bit has no effect on filtering of instructions.
0b1	If PMICFILTR_EL0.NSH == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.NSH == 1, then instructions in Secure EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When !IsSecureEL2Enabled(), access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

T, bit [23]

When FEAT_TME is implemented:

Non-transactional state filtering. Controls counting instructions in Non-transactional state.

T	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in Non-transactional state are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 filtering. Controls counting instructions in Realm EL1. If PMICFILTR_EL0.RLK is equal to PMICFILTR_EL0.P, then instructions in Realm EL1 are counted. Otherwise, instructions in Realm EL1 are not counted.

RLK	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in Realm EL1 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in Realm EL1 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 filtering. Controls counting instructions in Realm EL0. If PMICFILTR_EL0.RLU is equal to PMICFILTR_EL0.U, then instructions in Realm EL0 are counted. Otherwise, instructions in Realm EL0 are not counted.

RLU	Meaning
0b0	If PMICFILTR_EL0.U == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.U == 1, then instructions in Realm EL0 are not counted.
0b1	If PMICFILTR_EL0.U == 0, then instructions in Realm EL0 are not counted. If PMICFILTR_EL0.U == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]**When FEAT_RME is implemented:**

Realm EL2 filtering. Controls counting instructions in Realm EL2. If PMICFILTR_EL0.RLH is not equal to PMICFILTR_EL0.NSH, then instructions in Realm EL2 are counted. Otherwise, instructions in Realm EL2 are not counted.

RLH	Meaning
0b0	If PMICFILTR_EL0.NSH == 0, then instructions in Realm EL2 are not counted. If PMICFILTR_EL0.NSH == 1, then this bit has no effect on filtering of instructions.
0b1	If PMICFILTR_EL0.NSH == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.NSH == 1, then instructions in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:16]

Reserved, RES0.

evtCount, bits [15:0]

Event to count.

Reads as 0x0008.

Access to this field is **RO**.

Accessing PMICFILTR_EL0

PMICFILTR_EL0 reads-as-zero and ignores writes if all of the following are true:

- PSTATE.EL == EL0.
- [PMUSERENR_EL0](#).{UEN,EN} == {1,0}.
- [PMUACR_EL1](#).F0 == 0.

PMICFILTR_EL0 ignores writes if all of the following are true:

- PSTATE.EL == EL0.
- [PMUSERENR_EL0](#).{UEN,IR,EN} == {1,1,0}.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMICFILTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b0110	0b000

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = PMICFILTR_EL0;
    elsif PSTATE.EL == EL1 then
        X[t, 64] = PMICFILTR_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMICFILTR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMICFILTR_EL0;

```

MSR PMICFILTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b0110	0b000

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            PMICFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        PMICFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMICFILTR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMICFILTR_EL0 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMICNTR_EL0, Performance Monitors Instruction Counter Register

The PMICNTR_EL0 characteristics are:

Purpose

If event counting is not prohibited and the instruction counter is enabled, the counter increments for each architecturally-executed instruction, according to the configuration specified by [PMICFILTR_EL0](#).

Configuration

AArch64 System register PMICNTR_EL0 bits [63:0] are architecturally mapped to External register [PMU.PMICNTR_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3_ICNTR is implemented. Otherwise, direct accesses to PMICNTR_EL0 are UNDEFINED.

Attributes

PMICNTR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ICNT															
																ICNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ICNT, bits [63:0]

Instruction Counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMICNTR_EL0

PMICNTR_EL0 reads-as-zero and ignores writes if all of the following are true:

- PSTATE.EL == EL0.
- [PMUSERENR_EL0](#).{UEN,EN} == {1,0}.
- [PMUACR_EL1](#).F0 == 0.

PMICNTR_EL0 ignores writes if all of the following are true:

- PSTATE.EL == EL0.
- [PMUSERENR_EL0](#).{UEN,IR,EN} == {1,1,0}.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMICNTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b0100	0b000

```

if PSTATE.EL == EL0 then
    X[t, 64] = PMICNTR_EL0;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMICNTR_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMICNTR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMICNTR_EL0;

```

MSR PMICNTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b0100	0b000

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        PMICNTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    PMICNTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMICNTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMICNTR_EL0 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMICNTSVR_EL1, Performance Monitors Instruction Count Saved Value Register

The PMICNTSVR_EL1 characteristics are:

Purpose

Captures the PMU Instruction counter, [PMICNTR_EL0](#).

Configuration

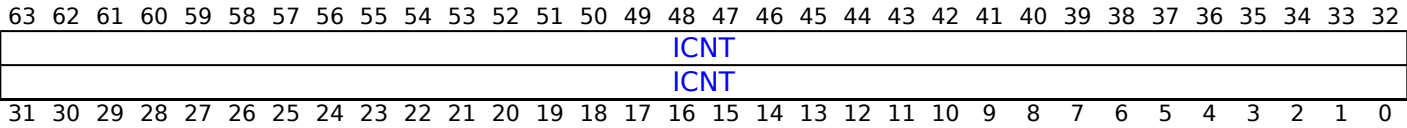
AArch64 System register PMICNTSVR_EL1 bits [63:0] are architecturally mapped to External register [PMU.PMICNTSVR_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_ICNTR is implemented and FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMICNTSVR_EL1 are UNDEFINED.

Attributes

PMICNTSVR_EL1 is a 64-bit register.

Field descriptions



ICNT, bits [63:0]

Sampled Instruction Count. The value of [PMICNTR_EL0](#) at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMICNTSVR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMICNTSVR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1110	0b1100	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMICNTSVR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMICNTSVR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMICNTSVR_EL1;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR_EL1 characteristics are:

Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR_EL0](#), and the event counters [PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

Configuration

AArch64 System register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENCLR\[31:0\]](#).

AArch64 System register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to External register [PMU.PMINTENCLR_EL1\[31:0\]](#)[PMINTENCLR_EL1\[31:0\]](#).

AArch64 System register PMINTENCLR_EL1 bits [63:32] are architecturally mapped to External register [PMU.PMINTENCLR_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMINTENCLR_EL1 are UNDEFINED.

Attributes

PMINTENCLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:[33](#)[32](#)]

Reserved, RES0.

F<m>, bit [m+32], for m = 0

When FEAT_PMUv3_ICNTR is implemented:

Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> disable. On writes, allows software to disable the interrupt request or PMU exception on unsigned overflow of fixed-function counter <m>. On reads, returns the interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> enable status.

F<m>	Meaning
0b0	Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> disabled.
0b1	Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> enabled.

PMINTENCLR_EL1.F0 holds the enable status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

[PMCCNTR_EL0](#) overflow interrupt request disable bit. Possible values are:

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request disable bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that the PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, disables the PMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENCLR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMINTENCLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMINTEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMINTENCLR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMINTENCLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMINTENCLR_EL1;

```

MSR PMINTENCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMINTEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMINTENCLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                PMINTENCLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMINTENCLR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMINTENSET_EL1, Performance Monitors Interrupt Enable Set register

The PMINTENSET_EL1 characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR_EL0](#), and the event counters [PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

Configuration

AArch64 System register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENSET\[31:0\]](#).

AArch64 System register PMINTENSET_EL1 bits [31:0] are architecturally mapped to External register [PMU.PMINTENSET_EL1\[31:0\]](#) [PMINTENSET_EL1\[31:0\]](#).

AArch64 System register PMINTENSET_EL1 bits [63:32] are architecturally mapped to External register [PMU.PMINTENSET_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMINTENSET_EL1 are UNDEFINED.

Attributes

PMINTENSET_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:**33**32]

Reserved, RES0.

F<m>, bit [m+32], for m = 0

When FEAT_PMUv3_ICNTR is implemented:

Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> enable. On writes, allows software to enable the interrupt request or PMU exception on unsigned overflow of fixed-function counter <m>. On reads, returns the interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> enable status.

F<m>	Meaning
0b0	Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> disabled.
0b1	Interrupt request or PMU exception on unsigned overflow of fixed-function counter <m> enabled.

PMINTENSET_EL1.F0 holds the enable status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **WIS**.

Otherwise:

Reserved, RES0.

C, bit [31]

[PMCCNTR_EL0](#) overflow interrupt request enable bit. Possible values are:

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request enable bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that the PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, enables the PMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENSET_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMINTENSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMINTEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMINTENSET_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMINTENSET_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMINTENSET_EL1;

```

MSR PMINTENSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMINTEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMINTENSET_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMINTENSET_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMINTENSET_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMMIR_EL1, Performance Monitors Machine Identification Register

The PMMIR_EL1 characteristics are:

Purpose

Describes Performance Monitors parameters specific to the implementation to software.

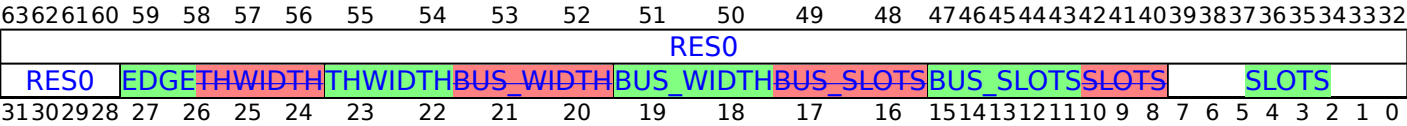
Configuration

This register is present only when FEAT_PMUv3p4 is implemented. Otherwise, direct accesses to PMMIR_EL1 are UNDEFINED.

Attributes

PMMIR_EL1 is a 64-bit register.

Field descriptions



Bits [63:2824]

Reserved, RES0.

EDGE, bits [27:24]

PMU event edge detection. Indicates implementation of the FEAT_PMUv3_EDGE feature.

EDGE	Meaning
0b0000	FEAT_PMUv3_EDGE is not implemented.
0b0001	FEAT_PMUv3_EDGE is implemented.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is RO.

THWIDTH, bits [23:20]

PMEVTYPEPER<n>_ELO.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the PMEVTYPEPER<n>_ELO.TH field.

THWIDTH	Meaning
0b0000	FEAT_PMUv3_TH is not implemented.
0b0001	1 bit. PMEVTYPER<n>_EL0.TH[11:1] are RES0.
0b0010	2 bits. PMEVTYPER<n>_EL0.TH[11:2] are RES0.
0b0011	3 bits. PMEVTYPER<n>_EL0.TH[11:3] are RES0.
0b0100	4 bits. PMEVTYPER<n>_EL0.TH[11:4] are RES0.
0b0101	5 bits. PMEVTYPER<n>_EL0.TH[11:5] are RES0.
0b0110	6 bits. PMEVTYPER<n>_EL0.TH[11:6] are RES0.
0b0111	7 bits. PMEVTYPER<n>_EL0.TH[11:7] are RES0.
0b1000	8 bits. PMEVTYPER<n>_EL0.TH[11:8] are RES0.
0b1001	9 bits. PMEVTYPER<n>_EL0.TH[11:9] are RES0.
0b1010	10 bits. PMEVTYPER<n>_EL0.TH[11:10] are RES0.
0b1011	11 bits. PMEVTYPER<n>_EL0.TH[11] is RES0.
0b1100	12 bits.

All other values are reserved.

If FEAT_PMUv3_TH is not implemented, this field is zero.

Otherwise, the largest value that can be written to [PMEVTYPER<n>_EL0.TH](#) is $2^{(\text{PMMIR_EL1.THWIDTH})}$ minus one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_WIDTH, bits [19:16]

Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\text{Log}_2(\text{number of bytes})$, plus one.

BUS_WIDTH	Meaning
0b0000	The information is not available.
0b0011	Four bytes.
0b0100	8 bytes.
0b0101	16 bytes.
0b0110	32 bytes.
0b0111	64 bytes.
0b1000	128 bytes.
0b1001	256 bytes.
0b1010	512 bytes.
0b1011	1024 bytes.
0b1100	2048 bytes.

All other values are reserved.

Each transfer is up to this number of bytes. An access might be smaller than the bus width.

When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.

When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS.

If the bus count information is not available, this field will read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SLOTS, bits [7:0]

Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMMIR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMMIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMMIR_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

PMOVSLR_EL0, Performance Monitors Overflow Flag Status Clear Register

The PMOVSLR_EL0 characteristics are:

Purpose

Contains the state of the overflow bit for the Cycle Count Register, [PMCCNTR_EL0](#), and each of the implemented event counters [PMEVCNTR<n>_EL0](#). Writing to this register clears these bits.

Configuration

AArch64 System register PMOVSLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMOVSRI31:0](#).

AArch64 System register PMOVSLR_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMOVSLR_EL0\[31:0\]](#)[PMOVSLR_EL0\[31:0\]](#).

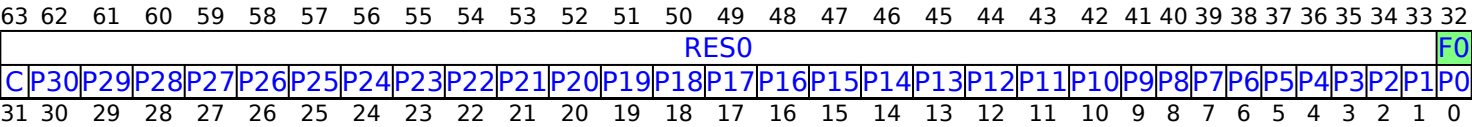
AArch64 System register PMOVSLR_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMOVSLR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMOVSLR_EL0 are UNDEFINED.

Attributes

PMOVSLR_EL0 is a 64-bit register.

Field descriptions



Bits [63:**33**32]

Reserved, RES0.

F<m>, bit [m+32], for m = 0
When FEAT_PMUv3_ICNTR is implemented:

Unsigned overflow flag for fixed-function counter <m> clear. On writes, allows software to clear the unsigned overflow flag for fixed-function counter <m> to 0. On reads, returns the unsigned overflow flag for fixed-function counter <m> overflow status.

F<m>	Meaning
0b0	Fixed-function counter <m> has not overflowed.
0b1	Fixed-function counter <m> has overflowed.

PMOVSLR_EL0.F0 holds the overflow status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

Cycle counter overflow clear bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.

[PMCR_EL0](#).LC controls whether an overflow is detected from unsigned overflow of [PMCCNTR_EL0](#)[31:0] or unsigned overflow of [PMCCNTR_EL0](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow clear bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, clears the PMEVCNTR<n>_EL0 overflow bit to 0.

If FEAT_PMuV3p5 is implemented, [MDCR_EL2](#).HLP and [PMCR_EL0](#).LP control whether an overflow is detected from unsigned overflow of [PMEVCNTR<n>_EL0](#)[31:0] or unsigned overflow of [PMEVCNTR<n>_EL0](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSLR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMOVSLR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSLR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSLR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSLR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMOVSLR_EL0;

```

MSR PMOVSLR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b011


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMOVSLR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMOVSSET_EL0, Performance Monitors Overflow Flag Status Set register

The PMOVSSET_EL0 characteristics are:

Purpose

Sets the state of the overflow bit for the Cycle Count Register, [PMCCNTR_EL0](#), and each of the implemented event counters [PMEVCNTR<n>_EL0](#).

Configuration

AArch64 System register PMOVSSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMOVSSETI\[31:0\]](#).

AArch64 System register PMOVSSET_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMOVSSET_EL0\[31:0\]](#)[PMOVSSET_EL0\[31:0\]](#).

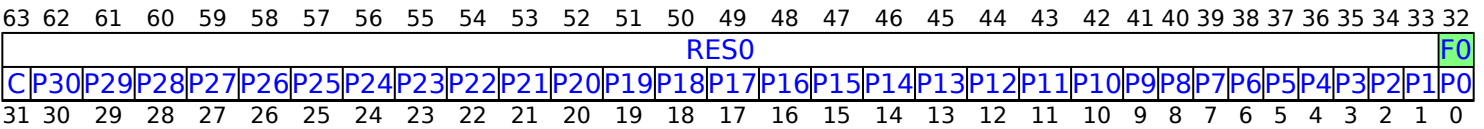
AArch64 System register PMOVSSET_EL0 bits [63:32] are architecturally mapped to External register [PMU.PMOVSSET_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMOVSSET_EL0 are UNDEFINED.

Attributes

PMOVSSET_EL0 is a 64-bit register.

Field descriptions



Bits [63:**33****32**]

Reserved, RES0.

F<m>, bit [m+32], for m = 0
When FEAT_PMUv3_ICNTR is implemented:

Unsigned overflow flag for fixed-function counter <m> set. On writes, allows software to set the unsigned overflow flag for fixed-function counter <m> to 1. On reads, returns the unsigned overflow flag for fixed-function counter <m> overflow status.

F<m>	Meaning
0b0	Fixed-function counter <m> has not overflowed.
0b1	Fixed-function counter <m> has overflowed.

PMOVSSET_EL0.F0 holds the overflow status for [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **W1S**.

Otherwise:

Reserved, RES0.

C, bit [31]

Cycle counter overflow set bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.

[PMCR_EL0](#).LC controls whether an overflow is detected from unsigned overflow of [PMCCNTR_EL0](#)[31:0] or unsigned overflow of [PMCCNTR_EL0](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow set bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, sets the PMEVCNTR<n>_EL0 overflow bit to 1.

If FEAT_PMuV3p5 is implemented, [MDCR_EL2](#).HLP and [PMCR_EL0](#).LP control whether an overflow is detected from unsigned overflow of [PMEVCNTR<n>_EL0](#)[31:0] or unsigned overflow of [PMEVCNTR<n>_EL0](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSSET_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMOVSSET_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSSET_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSSET_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMOVSSET_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMOVSSET_EL0;

```

MSR PMOVSSET_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMOVS == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMOVSSET_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMOVSSET_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSCR_EL1, Statistical Profiling Control Register (EL1)

The PMSCR_EL1 characteristics are:

Purpose

Provides EL1 controls for Statistical Profiling.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSCR_EL1 are UNDEFINED.

Attributes

PMSCR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																								PCT	TS	PA	CX	RES0	E1SPE	E0SPE	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:8]

Reserved, RES0.

PCT, bits [7:6]

When EL2 is implemented:

Physical Timestamp. If timestamp sampling is enabled and the Profiling Buffer is owned by EL1, requests which timestamp counter value is collected.

If FEAT_ECV is implemented, this is a two-bit field as shown. Otherwise, bit[7] is RES0.

PCT	Meaning	Applies when
0b00	Virtual timestamp. The collected timestamp is the physical counter minus the value of CNTVOFF_EL2 .	
0b01	Physical timestamp. The collected timestamp is the physical counter.	
0b11	Guest physical timestamp. The collected timestamp is the physical counter minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTPOFF_EL2 : <ul style="list-style-type: none"> SCR_EL3.ECVEN == 0. CNTHCTL_EL2.ECV == 0. 	When FEAT_ECV is implemented

If EL2 is enabled in the current Security state, then the value of [PMSCR_EL2.PCT](#) might override or modify the meaning of this field.

This field is ignored by the PE when the Profiling Buffer owning Exception level is EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Physical Timestamp. Reserved. This field reads as 0b01 and ignores writes. Software should treat this field as UNK/SBZP.

When EL2 is not implemented, the Effective values of [CNTVOFF_EL2](#) and [CNTPOFF_EL2](#) are zero, meaning the virtual counter and physical counter have the same value.

TS, bit [5]

Timestamp enable.

TS	Meaning
0b0	Timestamp sampling disabled.
0b1	Timestamp sampling enabled.

This bit is ignored by the PE if EL2 is implemented and the Profiling Buffer is owned by EL2. For more information, see 'Controlling the data that is collected'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PA, bit [4]

Physical Address sample enable.

PA	Meaning
0b0	Physical addresses are not collected.
0b1	Physical addresses are collected.

If EL2 is implemented:

- If the Profiling Buffer is owned by EL1, this bit is combined with [PMSCR_EL2.PA](#) to determine which address is collected. For more information, see 'Controlling the data that is collected'.
- If the Profiling Buffer is owned by EL2, this bit is ignored by the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CX, bit [3]

[CONTEXTIDR_EL1](#) sample enable.

CX	Meaning
0b0	CONTEXTIDR_EL1 is not collected.
0b1	CONTEXTIDR_EL1 is collected.

If EL2 is implemented and enabled in the current Security state when an operation is sampled:

- If the PE is at EL2, this bit is ignored by the PE.
- If [HCR_EL2.TGE](#) == 1, this bit is ignored by the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [2]

Reserved, RES0.

E1SPE, bit [1]

EL1 Statistical Profiling Enable.

E1SPE	Meaning
0b0	Sampling disabled at EL1.
0b1	Sampling enabled at EL1.

If EL2 is implemented and enabled in the current Security state, this bit is ignored by the PE when [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E0SPE, bit [0]

EL0 Statistical Profiling Enable. Controls sampling at EL0 when [HCR_EL2.TGE](#) == 0 or if EL2 is disabled or not implemented.

E0SPE	Meaning
0b0	Sampling disabled at EL0.
0b1	Sampling enabled at EL0.

If EL2 is implemented and enabled in the current Security state, this bit is ignored by the PE when [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x828];
    else
        X[t, 64] = PMSCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PMSCR_EL2;
    else
        X[t, 64] = PMSCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSCR_EL1;

```

MSR PMSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x828] = X[t, 64];
        else
            PMSCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            PMSCR_EL2 = X[t, 64];
        else
            PMSCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMSCR_EL1 = X[t, 64];

```

MRS <Xt>, PMSCR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x828];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = PMSCR_EL1;
    else
        UNDEFINED;

```

MSR PMSCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x828] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        PMSCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSCR_EL2, Statistical Profiling Control Register (EL2)

The PMSCR_EL2 characteristics are:

Purpose

Provides EL2 controls for Statistical Profiling.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSCR_EL2 are UNDEFINED.

Attributes

PMSCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																								PCT	TS	PA	CX	RES0	E2SPE	E0HSPE	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:8]

Reserved, RES0.

PCT, bits [7:6]

Physical Timestamp. If timestamp sampling is enabled, determines which counter is collected. The behavior depends on the Profiling Buffer owning Exception level.

If FEAT_ECV is implemented, this is a two-bit field as shown. Otherwise, bit[7] is RES0.

PCT	Meaning	Applies when
0b00	<p>Virtual timestamp. The collected timestamp is the physical counter minus a virtual offset. If any of the following are true, the virtual offset is zero, otherwise the virtual offset is the value of CNTVOFF_EL2:</p> <ul style="list-style-type: none"> The sampled operation executed at EL2 and HCR_EL2.E2H == 1. The sampled operation executed at EL0 and HCR_EL2.{E2H,TGE} == {1,1}. 	
	<p>Note If the Profiling Buffer owning Exception level is EL1, the virtual offset is always CNTVOFF_EL2.</p>	
0b01	<p>If the Profiling Buffer owning Exception level is EL1, then the timestamp value is selected by PMSCR_EL1.PCT. Otherwise, physical timestamp. The collected timestamp is the physical counter.</p>	
0b11	<p>If the Profiling Buffer owning Exception level is EL1 and PMSCR_EL1.PCT == 0b00, then guest virtual timestamp. The collected timestamp is the physical counter minus the value of CNTVOFF_EL2. Otherwise, guest physical timestamp. The collected timestamp is the physical counter minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTPOFF_EL2:</p> <ul style="list-style-type: none"> SCR_EL3.ECVEn == 0. CNTHCTL_EL2.ECV == 0. 	When FEAT_ECV is implemented

All other values are reserved.

If EL2 is not implemented or EL2 is disabled in the current Security state, then the Effective value of this field is 0b01, other than for a direct read of the register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TS, bit [5]

Timestamp Enable.

TS	Meaning
0b0	Timestamp sampling disabled.
0b1	Timestamp sampling enabled.

This bit is ignored by the PE when any of the following are true:

- The Profiling Buffer owning Exception level is EL1.
- In Secure state, and either FEAT_SEL2 is not implemented or Secure EL2 is disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PA, bit [4]

Physical Address Sample Enable.

PA	Meaning
0b0	Physical addresses are not collected.
0b1	Physical addresses are collected.

If the Profiling Buffer owning Exception level is EL1, and EL2 is enabled in the current Security state, this bit is combined with [PMSCR_EL1.PA](#) to determine which address is collected.

If EL2 is not implemented or EL2 is disabled in the current Security state, the PE ignores the value of this bit and behaves as if this bit is set to 1, other than for a direct read of the register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CX, bit [3]

[CONTEXTIDR_EL2](#) Sample Enable.

CX	Meaning
0b0	CONTEXTIDR_EL2 is not collected.
0b1	CONTEXTIDR_EL2 is collected.

If EL2 is not implemented or EL2 is disabled in the current Security state, the PE ignores the value of this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [2]

Reserved, RES0.

E2SPE, bit [1]

EL2 Statistical Profiling Enable.

E2SPE	Meaning
0b0	Sampling disabled at EL2.
0b1	Sampling enabled at EL2.

This bit is RES0 if [MDCR_EL2.E2PB](#) != 0b00.

If EL2 is disabled in the current Security state, this bit is ignored by the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E0HSPE, bit [0]

EL0 Statistical Profiling Enable.

E0HSPE	Meaning
0b0	Sampling disabled at EL0.
0b1	Sampling enabled at EL0.

If [MDCR_EL2.E2PB](#) != 0b00, this bit is RES0.

If EL2 is implemented and enabled in the current Security state, this bit is ignored by the PE when [HCR_EL2.TGE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSCR_EL2;

```

MSR PMSCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSCR_EL2 = X[t, 64];

```


MRS <Xt>, PMSCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x828];
    else
        X[t, 64] = PMSCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = PMSCR_EL2;
    else
        X[t, 64] = PMSCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSCR_EL1;

```

MSR PMSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x828] = X[t, 64];
    else
        PMSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        PMSCR_EL2 = X[t, 64];
    else
        PMSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMSDSFR_EL1, Sampling Data Source Filter Register

The PMSDSFR_EL1 characteristics are:

Purpose

Controls sample filtering by Data Source.

Configuration

This register is present only when FEAT_SPE_FDS is implemented. Otherwise, direct accesses to PMSDSFR_EL1 are UNDEFINED.

Attributes

PMSDSFR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
S63	S62	S61	S60	S59	S58	S57	S56	S55	S54	S53	S52	S51	S50	S49	S48	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35
S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3

S<m>, bit [m], for m = 63 to 0
When filtering on Data Source <m> is supported:

S[<m>] is the Data Source filter for IMPLEMENTATION DEFINED Data Source <m>.

S<m>	Meaning
0b0	If PMSFCR_EL1 .FDS is 1, do not record load operations that have bits [5:0] of the Data Source packet set to <m>.
0b1	Load operations with Data Source <m> are unaffected by PMSFCR_EL1 .FDS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

Accessing PMSDSFR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSDSFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x858];
    else
        X[t, 64] = PMSDSFR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSDSFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSDSFR_EL1;

```

MSR PMSDSFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x858] = X[t, 64];
    else
        PMSDSFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSDSFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSDSFR_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSELR_EL0, Performance Monitors Event Counter Selection Register

The PMSELR_EL0 characteristics are:

Purpose

Selects the current event counter [PMEVCNTR<n>_EL0](#) or the cycle counter, CCNT.

PMSELR_EL0 is used in conjunction with [PMXEVTYPER_EL0](#) to determine the event that increments a selected event counter, and the modes and states in which the selected counter increments.

It is also used in conjunction with [PMXVCNTR_EL0](#), to determine the value of a selected event counter.

Configuration

AArch64 System register PMSELR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMSELR\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMSELR_EL0 are UNDEFINED.

Attributes

PMSELR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL																															

Bits [63:5]

Reserved, RES0.

SEL, bits [4:0]

Selects event counter, [PMEVCNTR<n>_EL0](#), where n is the value held in this field. This value identifies which event counter is accessed when a subsequent access to [PMXEVTYPER_EL0](#) or [PMXVCNTR_EL0](#) occurs.

This field can take any value from 0 (0b00000) to (PMCR.N)-1, or 31 (0b11111).

When PMSELR_EL0.SEL is 0b11111, it selects the cycle counter and:

- A read of the [PMXEVTYPER_EL0](#) returns the value of [PMCCFILTR_EL0](#).
- A write of the [PMXEVTYPER_EL0](#) writes to [PMCCFILTR_EL0](#).
- A read or write of [PMXVCNTR_EL0](#) has CONSTRAINED UNPREDICTABLE effects. For more information, see [PMXVCNTR_EL0](#).

For more information about the results of accesses to the event counters, see [PMXEVTYPER_EL0](#) and [PMXVCNTR_EL0](#).

For more information about the number of counters accessible at each Exception level, see [MDCR_EL2](#).HPMN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSELR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSELR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMSELR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSELR_EL0;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSELR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSELR_EL0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSELR_EL0;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMSELR_EL0;

```

MSR PMSELR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMSELR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSELR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSELR_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSELR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSELR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMSELR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165fb91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSEVFR_EL1, Sampling Event Filter Register

The PMSEVFR_EL1 characteristics are:

Purpose

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if PMSEVFR_EL1.E[3] and PMSEVFR_EL1.E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 (TLB walk) set to 1 are recorded.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSEVFR_EL1 are UNDEFINED.

Attributes

PMSEVFR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49						
E[63]	E[62]	E[61]	E[60]	E[59]	E[58]	E[57]	E[56]	E[55]	E[54]	E[53]	E[52]	E[51]	E[50]	E[49]						
E[31]	E[30]	E[29]	E[28]	E[27]	E[26]	E[25]	E[24]	E[23] RAZ/WI	E[22]	E[18]	E[21]	E[17]	E[20]	E[16]	E[19]	E[15]	E[18]	E[14]	E[17]	E[13]
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11

E[<x>], bit [x], for x = 63 to 48, 31 to 24, 15 to 12

E[<x>] is the event filter for event <x>. If event <x> is not implemented, or filtering on event <x> is not supported, the corresponding bit is RAZ/WI.

E[<x>]	Meaning
0b0	Event <x> is ignored.
0b1	Do not record samples that have event <x> == 0.

An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.

This field is ignored by the PE when PMSFCR_EL1.FE == 0

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [47:32]

Reserved, RAZ/WI.

E[23], Bits bit [23:19]
When FEAT_SPEv1p4 is implemented and event 23 is implemented:

Data snooped.

E[23]	Meaning
0b0	Data snooped event is ignored.
0b1	Do not record samples that have the Data snooped event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[22], bit [22]

When FEAT_SPEv1p4 is implemented and event 22 is implemented:

Recently fetched.

E[22]	Meaning
0b0	Recently fetched event is ignored.
0b1	Do not record samples that have the Recently fetched event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[21], bit [21]

When FEAT_SPEv1p4 is implemented and event 21 is implemented:

Cache data modified.

E[21]	Meaning
0b0	Cache data modified event is ignored.
0b1	Do not record samples that have the Cache data modified event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[20], bit [20]

When FEAT_SPEv1p4 is implemented and event 20 is implemented:

Level 2 data cache miss.

E[20]	Meaning
0b0	Level 2 data cache miss event is ignored.
0b1	Do not record samples that have the Level 2 data cache miss event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[19], bit [19]

When FEAT_SPEv1p4 is implemented and event 19 is implemented:

Level 2 data cache access.

E[19]	Meaning
0b0	Level 2 data cache access event is ignored.
0b1	Do not record samples that have the Level 2 data cache access event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[18], bit [18]

When FEAT_SPEv1p1 is implemented and FEAT_SVE is implemented:

Empty predicate.

E[18]	Meaning
0b0	Empty predicate event is ignored.
0b1	Do not record samples that have the Empty predicate event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[17], bit [17]

When FEAT_SPEv1p1 is implemented and FEAT_SVE is implemented:

Partial predicate.

E[17]	Meaning
0b0	Partial predicate event is ignored.
0b1	Do not record samples that have the Partial predicate event == 0.

This bit is ignored by the PE when [PMSFCR_EL1](#).FE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[16], bit [16]

When FEAT_TME is implemented:

Transactional

E[16]	Meaning
0b0	Transactional event is ignored.
0b1	Do not record samples that have the Transactional event == 0.

This bit is ignored by the PE when [PMSFCR_EL1](#).FE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[11], bit [11]

When FEAT_SPEv1p1 is implemented:

Alignment.

E[11]	Meaning
0b0	Alignment event is ignored.
0b1	Do not record samples that have the Alignment event == 0.

This bit is ignored by the PE when [PMSFCR_EL1](#).FE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[10], Bits bit [10:8]

When (FEAT_SPEv1p4 is implemented or filtering on event 10 is optionally supported) and event 10 is implemented:

Remote access.

E[10]	Meaning
0b0	Remote access event is ignored.
0b1	Do not record samples that have the Remote access event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[9], bit [9]

When (FEAT_SPEv1p4 is implemented or filtering on event 9 is optionally supported) and event 9 is implemented:

Last Level cache miss.

E[9]	Meaning
0b0	Last Level cache miss event is ignored.
0b1	Do not record samples that have the Last Level cache miss event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[8], bit [8]

When (FEAT_SPEv1p4 is implemented or filtering on event 8 is optionally supported) and event 8 is implemented:

Last Level cache access.

E[8]	Meaning
0b0	Last Level cache access event is ignored.
0b1	Do not record samples that have the Last Level cache access event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[7], bit [7]

Mispredicted.

E[7]	Meaning
0b0	Mispredicted event is ignored.
0b1	Do not record samples that have the Mispredicted event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[6], bit [6]**When FEAT_SPEv1p2 is implemented:**

Not taken.

E[6]	Meaning
0b0	Not taken event is ignored.
0b1	Do not record samples that have the Not taken event == 0.

This field is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[5], bit [5]

TLB walk.

E[5]	Meaning
0b0	TLB walk event is ignored.
0b1	Do not record samples that have the TLB walk event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[4], bit [4]**When FEAT_SPEv1p4 is implemented or filtering on event 4 is optionally supported:**

TLB access.

E[4]	Meaning
0b0	TLB access event is ignored.
0b1	Do not record samples that have the TLB access event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[3], bit [3]

Level 1 data or unified cache refill.

E[3]	Meaning
0b0	Level 1 data or unified cache refill event is ignored.
0b1	Do not record samples that have the Level 1 data or unified cache refill event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[2], bit [2]**When FEAT_SPEv1p4 is implemented or filtering on event 2 is optionally supported:**

Level 1 data cache access.

E[2]	Meaning
0b0	Level 1 data cache access event is ignored.
0b1	Do not record samples that have the Level 1 data cache access event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[1], bit [1]**When the PE supports sampling of speculative instructions:**

Architecturally executed.

When the PE supports sampling of speculative instructions:

E[1]	Meaning
0b0	Architecturally executed event is ignored.
0b1	Do not record samples that have the Architecturally executed event == 0.

This bit is ignored by the PE when [PMSFCR_EL1.FE](#) == 0.

If the PE does not support the sampling of speculative instructions, or always discards the sample record for speculative instructions, this bit reads as an UNKNOWN value and the PE ignores its value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, UNKNOWN.

Bit [0]

Reserved, RAZ/WI.

Accessing PMSEVFR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x830];
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSEVFR_EL1;

```

MSR PMSEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x830] = X[t, 64];
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSEVFR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMSFCR_EL1, Sampling Filter Control Register

The PMSFCR_EL1 characteristics are:

Purpose

Controls sample filtering. The filter is the logical AND of the FL, FT and FE bits. For example, if FE == 1 and FT == 1 only samples including the selected operation types and the selected events will be recorded

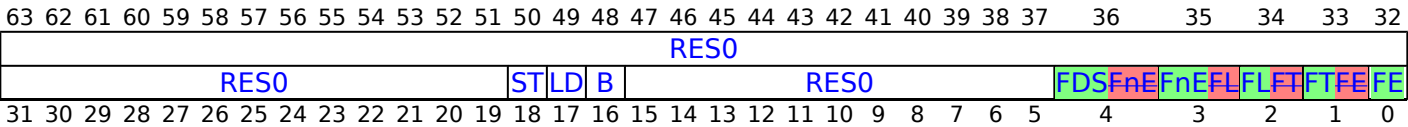
Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSFCR_EL1 are UNDEFINED.

Attributes

PMSFCR_EL1 is a 64-bit register.

Field descriptions



Bits [63:19]

Reserved, RES0.

ST, bit [18]

Store filter enable

ST	Meaning
0b0	Do not record store operations
0b1	Record all store operations, including vector stores and all atomic operations

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

LD, bit [17]

Load filter enable

LD	Meaning
0b0	Do not record load operations
0b1	Record all load operations, including vector loads and atomic operations that return data

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

B, bit [16]

Branch filter enable

B	Meaning
0b0	Do not record branch and exception return operations
0b1	Record all branch and exception return operations

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [15:54]

Reserved, RES0.

FDS, bit [4]**When FEAT_SPE_FDS is implemented:**

Filter by Data Source.

FDS	Meaning
0b0	Data Source filtering disabled.
0b1	Data Source filtering enabled. Samples of load instructions reporting a Data Source selected by PMSDSFR_EL1 will not be recorded.

If PMSFCR_EL1.FDS == 1 and PMSDSFR_EL1 is zero, then no load operations with a Data Source will be recorded.

Load operations without a Data Source and other sampled operations are unaffected by this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FnE, bit [3]**When FEAT_SPEv1p2 is implemented:**

Filter by event, inverted.

FnE	Meaning
0b0	Inverted event filtering disabled.
0b1	Inverted event filtering enabled. Samples including the events selected by PMSNEVFR_EL1 will not be recorded.

If any of the following are true, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FnE == 0:

- PMSFCR_EL1.FnE == 1 and PMSNEVFR_EL1 is zero.
- PMSFCR_EL1.FnE == 1, PMSFCR_EL1.FE == 1, and there exists a value x such that PMSEVFR_EL1.E[x] == 1 and PMSNEVFR_EL1.E[x] == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FL, bit [2]

Filter by latency

FL	Meaning
0b0	Latency filtering disabled
0b1	Latency filtering enabled. Samples with a total latency less than PMSLATFR_EL1.MINLAT will not be recorded

If this field is set to 1 and PMSLATFR_EL1.MINLAT is set to zero, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FL is set to 0

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FT, bit [1]

Filter by operation type. The filter is the logical OR of the ST, LD and B bits. For example, if LD and ST are both set, both load and store operations are recorded

FT	Meaning
0b0	Type filtering disabled
0b1	Type filtering enabled. Samples not one of the selected operation types will not be recorded

If this field is set to 1 and the PMSFCR_EL1.{ST, LD, B} bits are all set to zero, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FT is set to 0

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FE, bit [0]

Filter by event.

FE	Meaning
0b0	Event filtering disabled.
0b1	Event filtering enabled. Samples not including the events selected by PMSEVER_EL1 will not be recorded.

If any of the following are true, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FE == 0:

- PMSFCR_EL1.FE == 1 and [PMSEVER_EL1](#) is zero.
- FEAT_SPEv1p2 is implemented, PMSFCR_EL1.FnE == 1, PMSFCR_EL1.FE == 1, and there exists a value x such that [PMSEVER_EL1](#).E[x] == 1 and [PMSNEVER_EL1](#).E[x] == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSFCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSFCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSFCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSFCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSFCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSFCR_EL1;

```

MSR PMSFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSFCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSFCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSICR_EL1, Sampling Interval Counter Register

The PMSICR_EL1 characteristics are:

Purpose

Software must write zero to PMSICR_EL1 before enabling sample profiling for a sampling session. Software must then treat PMSICR_EL1 as an opaque, 64-bit, read/write register used for context switches only.

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSICR_EL1 are UNDEFINED.

The value of PMSICR_EL1 does not change whilst profiling is disabled.

Attributes

PMSICR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ECOUNT								RES0																							
COUNT																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ECOUNT, bits [63:56]

When PMSIDR_EL1.ERnd == 1:

Secondary sample interval counter.

This field provides the secondary counter used after the primary counter reaches zero. Whilst the secondary counter is nonzero and profiling is enabled, the secondary counter decrements by 1 for each member of the sample population. The primary counter also continues to decrement since it is also nonzero. When the secondary counter reaches zero, a member of the sampling population is selected for sampling.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [55:32]

Reserved, RES0.

COUNT, bits [31:0]

Primary sample interval counter

Provides the primary counter used for sampling.

The primary counter is reloaded when the value of this register is zero and the PE moves from a state or Exception level where profiling is disabled to a state or Exception level where profiling is enabled

Whilst the primary counter is nonzero and sampling is enabled, the primary counter decrements by 1 for each member of the sample population

When the counter reaches zero, the behavior depends on the values of PMSIDR_EL1.ERnd and PMSIRR_EL1.RND

- If [PMSIRR_EL1](#).RND == 0 or PMSIDR_EL1.ERnd == 0:
 - A member of the sampling population is selected for sampling
 - The primary counter is reloaded
- If [PMSIRR_EL1](#).RND == 1 and [PMSIDR_EL1](#).ERnd == 1:
 - The secondary counter is set to a random or pseudorandom value in the range 0x00 to 0xFF
 - The primary counter is reloaded

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSICR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSICR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSICR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x838];
    else
        X[t, 64] = PMSICR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSICR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSICR_EL1;

```

MSR PMSICR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSICR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x838] = X[t, 64];
    else
        PMSICR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSICR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSICR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The PMSIDR_EL1 characteristics are:

Purpose

Describes the Statistical Profiling implementation to software

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSIDR_EL1 are UNDEFINED.

Attributes

PMSIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																																
RES0						PBT	Format				CountSize				MaxSize				Interval				FDS	RES0	FnE	ERnd	LDS	ArchInst		FL	FT	FE
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:25]

Reserved, RES0.

PBT, bit [24]

Previous branch target Address packet. Defined values are:

PBT	Meaning
0b0	Previous branch target Address packet not supported.
0b1	Previous branch target Address packet support implemented.

FEAT_SPEv1p2 adds the OPTIONAL functionality identified by the value 1.

Format, bits [23:20]
From Armv8.7:

Defines the format of the sample records. Defined values are:

Format	Meaning
0b0000	Format 0.

All other values are reserved.

Otherwise:

Reserved, RAZ.

CountSize, bits [19:16]

Defines the size of the counters.

CountSize	Meaning	Applies when
0b0010	12-bit saturating counters.	
0b0011	16-bit saturating counters.	From Armv8.8

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

MaxSize, bits [15:12]

Defines the largest size for a single record, rounded up to a power-of-two. If this is the same as the minimum alignment ([PMBIDR_EL1.Align](#)), then each record is exactly this size. Defined values are:

MaxSize	Meaning
0b0100	16 bytes.
0b0101	32 bytes.
0b0110	64 bytes.
0b0111	128 bytes.
0b1000	256 bytes.
0b1001	512 bytes.
0b1010	1KB.
0b1011	2KB.

All other values are reserved.

The values 0b0100 and 0b0101 are not permitted for an implementation.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Interval, bits [11:8]

Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. Defined values are:

Interval	Meaning
0b0000	256.
0b0010	512.
0b0011	768.
0b0100	1,024.
0b0101	1,536.
0b0110	2,048.
0b0111	3,072.
0b1000	4,096.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

FDS, bitBit [7]

Filter by data source. Defined values are:

Reserved, RES0.

FDS	Meaning
0b0	PMSDSFR_EL1 is not implemented and PMSFCR_EL1.FDS is RES0.
0b1	PMSDSFR_EL1 and PMSFCR_EL1.FDS are implemented.

[FEAT_SPE_FDS](#) adds the functionality identified by the value 1.

FnE, bit [6]

Filtering by events, inverted. Defined values are:

FnE	Meaning
0b0	PMSNEVER_EL1 is not implemented and PMSFCR_EL1.FnE is RES0.
0b1	PMSNEVER_EL1 and PMSFCR_EL1.FnE are implemented.

[FEAT_SPEv1p2](#) adds the functionality identified by the value 1.

ERnd, bit [5]

Defines how the random number generator is used in determining the interval between samples, when enabled by [PMSIRR_EL1.RND](#). Defined values are:

ERnd	Meaning
0b0	The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires.
0b1	The random number is added and the new interval started after the interval programmed in PMSIRR_EL1.INTERVAL expires, and the sample is taken when the random interval expires.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

LDS, bit [4]

Data source indicator for sampled load instructions. Defined values are:

LDS	Meaning
0b0	Loaded data source not implemented.
0b1	Loaded data source implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

ArchInst, bit [3]

Architectural instruction profiling. Defined values are:

ArchInst	Meaning
0b0	Micro-op sampling implemented.
0b1	Architecture instruction sampling implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

FL, bit [2]

Filtering by latency. This bit is RAO.

FT, bit [1]

Filtering by operation type. This bit is RAO.

FE, bit [0]

Filtering by events. This bit is RAO.

Accessing PMSIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSIDR_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elseif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSIDR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = PMSIDR_EL1;

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMSIRR_EL1, Sampling Interval Reload Register

The PMSIRR_EL1 characteristics are:

Purpose

Defines the interval between samples.

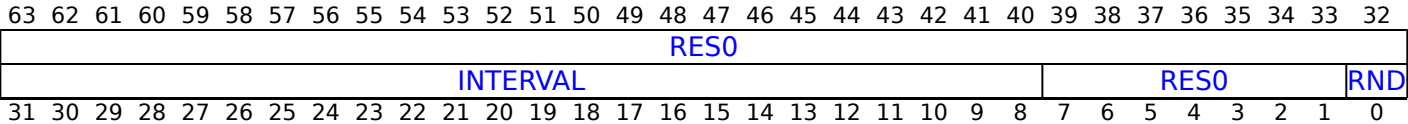
Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSIRR_EL1 are UNDEFINED.

Attributes

PMSIRR_EL1 is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, RES0.

INTERVAL, bits [31:8]

Bits [31:8] of the PMSICR_EL1 interval counter reload value. Software must set this to a non-zero value. If software sets this to zero, an UNKNOWN sampling interval is used. Software should set this to a value greater than the minimum indicated by PMSIDR_EL1.Interval.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:1]

Reserved, RES0.

RND, bit [0]

Controls randomization of the sampling interval.

RND	Meaning
0b0	Disable randomization of sampling interval.
0b1	Add (pseudo-)random jitter to sampling interval.

The random number generator is not architected.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSIRR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSIRR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSIRR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x840];
    else
        X[t, 64] = PMSIRR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIRR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSIRR_EL1;

```

MSR PMSIRR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSIRR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x840] = X[t, 64];
    else
        PMSIRR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSIRR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSIRR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSLATFR_EL1, Sampling Latency Filter Register

The PMSLATFR_EL1 characteristics are:

Purpose

Controls sample filtering by latency

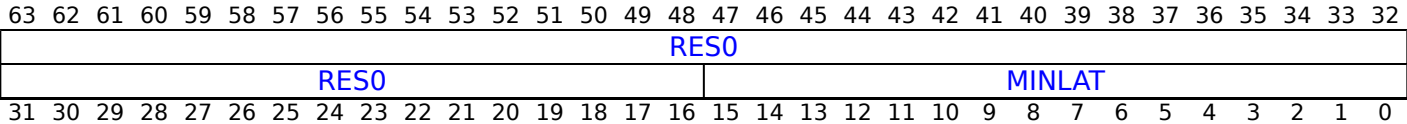
Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSLATFR_EL1 are UNDEFINED.

Attributes

PMSLATFR_EL1 is a 64-bit register.

Field descriptions



Bits [63:16]

Reserved, RES0.

MINLAT, bits [15:0]

Minimum latency. When [PMSFCR_EL1](#).FL is 1, defines the minimum total latency for filtered operations. Samples with a total latency less than PMSLATFR_EL1.MINLAT are not recorded.

If [PMSIDR_EL1](#).CountSize is 0b0010, PMSLATFR_EL1.MINLAT[15:12] is RES0.

This field is ignored by the PE when [PMSFCR_EL1](#).FL == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSLATFR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSLATFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMSLATFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x848];
    else
        X[t, 64] = PMSLATFR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSLATFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSLATFR_EL1;

```

MSR PMSLATFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSLATFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x848] = X[t, 64];
    else
        PMSLATFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSLATFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSLATFR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSNEVFR_EL1, Sampling Inverted Event Filter Register

The PMSNEVFR_EL1 characteristics are:

Purpose

Controls sample filtering by events. The overall inverted filter is the logical OR of these filters. For example, if PMSNEVFR_EL1.E[3] and PMSNEVFR_EL1.E[5] are both set to 1, samples that have either event 3 (Level 1 unified or data cache refill) or event 5 (TLB walk) set to 1 are not recorded.

Configuration

This register is present only when FEAT_SPEv1p2 is implemented. Otherwise, direct accesses to PMSNEVFR_EL1 are UNDEFINED.

Attributes

PMSNEVFR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
E[63]	E[62]	E[61]	E[60]	E[59]	E[58]	E[57]	E[56]	E[55]	E[54]	E[53]	E[52]	E[51]	E[50]	E[49]
E[31]	E[30]	E[29]	E[28]	E[27]	E[26]	E[25]	E[24]	E[23]	E[22]	E[21]	E[20]	E[19]	E[18]	E[17]
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17

E[<x>], bit [x], for x = 63 to 48, 31 to 24, 15 to 12

E[<x>] is the event filter for IMPLEMENTATION DEFINED event <x>.

E[<x>]	Meaning
0b0	Event <x> is ignored.
0b1	Do not record samples that have event <x> == 1.

An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, the corresponding bits of PMSNEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When event <x> is not implemented, or filtering on event <x> is not supported, access to this field is **RAZ/WI**.

Bits [47:32]

Reserved, RAZ/WI.

E[23], Bits bit [23:19]**When FEAT_SPEv1p4 is implemented and event 23 is implemented:**

Data not snooped.

E[23]	Meaning
0b0	Data snooped event is ignored.
0b1	Do not record samples that have the Data snooped event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[22], bit [22]**When FEAT_SPEv1p4 is implemented and event 22 is implemented:**

Not recently fetched.

E[22]	Meaning
0b0	Recently fetched event is ignored.
0b1	Do not record samples that have the Recently fetched event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[21], bit [21]**When FEAT_SPEv1p4 is implemented and event 21 is implemented:**

Cache data not modified.

E[21]	Meaning
0b0	Cache data modified event is ignored.
0b1	Do not record samples that have the Cache data modified event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[20], bit [20]**When FEAT_SPEv1p4 is implemented and event 20 is implemented:**

Level 2 data cache hit.

E[20]	Meaning
0b0	Level 2 data cache miss event is ignored.
0b1	Do not record samples that have the Level 2 data cache miss event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[19], bit [19]**When FEAT_SPEv1p4 is implemented and event 19 is implemented:**

No level 2 data cache access.

E[19]	Meaning
0b0	Level 2 data cache access event is ignored.
0b1	Do not record samples that have the Level 2 data cache access event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[18], bit [18]**When FEAT_SVE is implemented and FEAT_SPEv1p1 is implemented:**

Not empty predicate.

E[18]	Meaning
0b0	Empty predicate event is ignored.
0b1	Do not record samples that have the Empty predicate event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[17], bit [17]**When FEAT_SVE is implemented and FEAT_SPEv1p1 is implemented:**

Not partial predicate.

E[17]	Meaning
0b0	Partial predicate event is ignored.
0b1	Do not record samples that have the Partial predicate event == 1.

This field is ignored by the PE when [PMSFCR_EL1](#).FnE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[16], bit [16]**When FEAT_TME is implemented:**

Not transactional.

E[16]	Meaning
0b0	Transactional event is ignored.
0b1	Do not record samples that have the Transactional event == 1.

This field is ignored by the PE when [PMSFCR_EL1](#).FnE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[11], bit [11]**When FEAT_SPEv1p1 is implemented:**

Aligned.

E[11]	Meaning
0b0	Misalignment event is ignored.
0b1	Do not record samples that have the Misalignment event == 1.

This field is ignored by the PE when [PMSFCR_EL1](#).FnE == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[10], Bits bit [10:8]

When (FEAT_SPEv1p4 is implemented or filtering on event 10 is optionally supported) and event 10 is implemented:

No remote access.

E[10]	Meaning
0b0	Remote access event is ignored.
0b1	Do not record samples that have the Remote access event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[9], bit [9]

When (FEAT_SPEv1p4 is implemented or filtering on event 9 is optionally supported) and event 9 is implemented:

Last Level cache hit.

E[9]	Meaning
0b0	Last Level cache miss event is ignored.
0b1	Do not record samples that have the Last Level cache miss event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[8], bit [8]

When (FEAT_SPEv1p4 is implemented or filtering on event 8 is optionally supported) and event 8 is implemented:

No Last Level cache access.

E[8]	Meaning
0b0	Last Level cache access event is ignored.
0b1	Do not record samples that have the Last Level cache access event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[7], bit [7]

Correctly predicted.

E[7]	Meaning
0b0	Mispredicted event is ignored.
0b1	Do not record samples that have the Mispredicted event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[6], bit [6]

Taken.

E[6]	Meaning
0b0	Not taken event is ignored.
0b1	Do not record samples that have the Not taken event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[5], bit [5]

TLB hit.

E[5]	Meaning
0b0	TLB walk event is ignored.
0b1	Do not record samples that have the TLB walk event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[4], bit [4]**When FEAT_SPEv1p4 is implemented or filtering on event 4 is optionally supported:**

No TLB access.

E[4]	Meaning
0b0	TLB access event is ignored.
0b1	Do not record samples that have the TLB access event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[3], bit [3]

Level 1 data or unified cache hit.

E[3]	Meaning
0b0	Level 1 data or unified cache refill event is ignored.
0b1	Do not record samples that have the Level 1 data or unified cache refill event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E[2], bit [2]**When FEAT_SPEv1p4 is implemented or filtering on event 2 is optionally supported:**

No Level 1 data cache access.

E[2]	Meaning
0b0	Level 1 data cache access event is ignored.
0b1	Do not record samples that have the Level 1 data cache access event == 1.

This bit is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

E[1], bit [1]**When the PE supports sampling of speculative instructions:**

Speculative.

E[1]	Meaning
0b0	Architecturally executed event is ignored.
0b1	Do not record samples that have the Architecturally executed event == 1.

This field is ignored by the PE when [PMSFCR_EL1.FnE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAZ/WI.

Bit [0]

Reserved, RAZ/WI.

Accessing PMSNEVFR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSNEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.nPMSNEVFR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.EnPMSN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
            X[t, 64] = NVMem[0x850];
        else
            X[t, 64] = PMSNEVFR_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.EnPMSN == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.EnPMSN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMSNEVFR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = PMSNEVFR_EL1;

```

MSR PMSNEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.EnPMSN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.nPMSNEVFR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.EnPMSN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
            NVMem[0x850] = X[t, 64];
        else
            PMSNEVFR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.EnPMSN == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE)) then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.EnPMSN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSNEVFR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMSNEVFR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMSSCR_EL1, Performance Monitors Snapshot Status and Capture Register

The PMSSCR_EL1 characteristics are:

Purpose

Holds status information about the captured counters and provides a mechanism for software to initiate a sample.

Configuration

AArch64 System register PMSSCR_EL1 bits [63:0] are architecturally mapped to External register [PMU.PMSSCR_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMSSCR_EL1 are UNDEFINED.

Attributes

PMSSCR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																NC															
RES0																SS															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

NC, bit [32]

No Capture. Indicates whether the PMU counters have been captured.

NC	Meaning
0b0	PMU counters captured.
0b1	PMU counters not captured.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Bits [31:1]

Reserved, RES0.

SS, bit [0]

Snapshot Capture and Status.

SS	Meaning
0b0	On a read: The Capture event has completed. On a write: Ignored.
0b1	On a read: The Capture event has not completed. On a write: Initiate a capture immediately.

It is CONSTRAINED UNPREDICTABLE whether a Capture event has completed if this field is modified when the Capture event is ongoing.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMSSCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMSSCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1101	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMSSCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMSSCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSSCR_EL1;
```

MSR PMSSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1101	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PMSSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMSSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSSCR_EL1 = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSWINC_EL0, Performance Monitors Software Increment register

The PMSWINC_EL0 characteristics are:

Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW_INCR'.

Configuration

AArch64 System register PMSWINC_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMSWINC\[31:0\]](#).

AArch64 System register PMSWINC_EL0 bits [31:0] are architecturally mapped to External register [PMU.PMSWINC_EL0\[31:0\]](#)[PMSWINC_EL0\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMSWINC_EL0 are UNDEFINED.

Attributes

PMSWINC_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:31]

Reserved, RES0.

P<n>, bit [n], for n = 30 to 0

Event counter software increment bit for [PMEVCNTR<n>_EL0](#).

If N is less than 31, then bits [30:N] are WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN. Otherwise, N is the value in [PMCR_EL0](#).N.

P<n>	Meaning
0b0	No action. The write to this bit is ignored.
0b1	If PMEVCNTR<n>_EL0 is enabled and configured to count the software increment event, increments PMEVCNTR<n>_EL0 by 1. If PMEVCNTR<n>_EL0 is disabled, or not configured to count the software increment event, the write to this bit is ignored.

Accessing PMSWINC_EL0

Accesses to this register use the following encodings in the System register encoding space:

MSR PMSWINC_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<SW,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMSWINC_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSWINC_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMSWINC_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSWINC_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMSWINC_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMSWINC_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMUACR_EL1, Performance Monitors User Access Control Register

The PMUACR_EL1 characteristics are:

Purpose

Enables or disables EL0 access to specific Performance Monitors.

Configuration

This register is present only when FEAT_PMUv3p9 is implemented. Otherwise, direct accesses to PMUACR_EL1 are UNDEFINED.

Attributes

PMUACR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

F<m>, bit [m+32], for m = 0

When FEAT_PMUv3_ICNTR is implemented:

EL0 access to fixed-function counter <m> enable.

F<m>	Meaning
0b0	EL0 access to fixed-function counter <m> disabled when PMUSERENR_EL0 .{UEN,EN} == {1,0}.
0b1	EL0 access to fixed-function counter <m> enabled when PMUSERENR_EL0 .{UEN,EN} == {1,0}.

This bit is ignored by the PE when any of the following are true:

- All of the following are true:
 - EL2 is disabled in the current Security state, EL2 is using AArch32, or [HCR_EL2](#).{E2H,TGE} != {1,1}.
 - EL1 is using AArch32.
- [PMUSERENR_EL0](#).{UEN,EN} != {1,0}.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing this field has the following behavior:

- Access is **RAZ/WI** if all of the following are true:

- EL2Enabled()
- EL3 is not implemented or SCR_EL3.FGTEn2 == 1
- HDFGRTR2_EL2.nPMICFILTR_ELO == 0
- PSTATE.EL == EL1
- HCR_EL2.E2H != 1
- HCR_EL2.TGE != 1
- Access is **RAZ/WI** if all of the following are true:
 - EL3 is implemented
 - MDCR_EL3.EnPM2 == 0
 - PSTATE.EL IN {EL2, EL1}
- Access is **RO** if all of the following are true:
 - EL2Enabled()
 - EL3 is not implemented or SCR_EL3.FGTEn2 == 1
 - HDFGWTR2_EL2.nPMICFILTR_ELO == 0
 - PSTATE.EL == EL1
 - HCR_EL2.E2H != 1
 - HCR_EL2.TGE != 1
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

C, bit [31]

EL0 access to [PMCCNTR_ELO](#) enable.

C	Meaning
0b0	EL0 access to PMCCNTR_ELO disabled when PMUSERENR_ELO .{UEN,EN} == {1,0}.
0b1	EL0 access to PMCCNTR_ELO enabled when PMUSERENR_ELO .{UEN,EN} == {1,0}.

This field is ignored by the PE when any of the following are true:

- All of the following are true:
 - EL2 is disabled in the current Security state, EL2 is using AArch32, or [HCR_EL2](#).{E2H,TGE} != {1,1}.
 - EL1 is using AArch32.
- [PMUSERENR_ELO](#).{UEN,EN} != {1,0}.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<m>, bit [m], for m = 30 to 0

EL0 access to [PMEVCNTR<m>_ELO](#) enable.

P<m>	Meaning
0b0	EL0 access to PMEVCNTR<m>_ELO disabled when PMUSERENR_ELO .{UEN,EN} == {1,0}.
0b1	EL0 access to PMEVCNTR<m>_ELO enabled when PMUSERENR_ELO .{UEN,EN} == {1,0}.

This bit is ignored by the PE when any of the following are true:

- All of the following are true:
 - EL2 is disabled in the current Security state, EL2 is using AArch32, or [HCR_EL2](#).{E2H,TGE} != {1,1}.
 - EL1 is using AArch32.
- [PMUSERENR_ELO](#).{UEN,EN} != {1,0}.

Accessing this bit has the following behavior:

- This bit reads-as-zero and ignores writes if any of the following are true:

- All of the following are true:
 - EL2 is implemented and enabled in the current Security state.
 - $m \geq \text{UInt}(\text{MDCR_EL2.HPMN})$.
 - Accessed at EL1.
- $m \geq \text{UInt}(\text{PMCR_EL0.N})$.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMUACR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMUACR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMUACR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMUACR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMUACR_EL1;
```

MSR PMUACR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PMUACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMUACR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMUACR_EL1 = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

The PMUSERENR EL0 characteristics are:

Purpose

Enables or disables EL0 access to the Performance Monitors.

Configuration

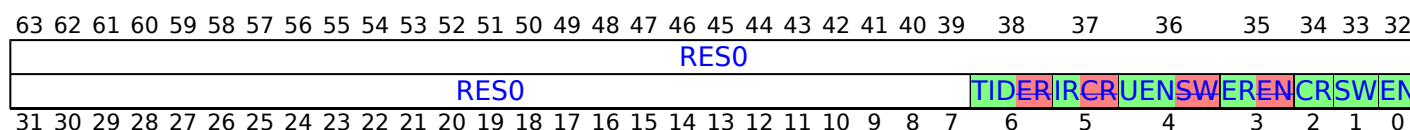
AArch64 System register PMUSERENR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMUSERENR\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMUSERENR_EL0 are UNDEFINED.

Attributes

PMUSERENR_EL0 is a 64-bit register.

Field descriptions



Bits [63:74]

Reserved, RES0.

TIDER, bit [63]

When FEAT PMUv3p9 is implemented:

In AArch64 state, trapped accesses are reported using EC syndrome value 0x18.

In AArch32 state, trapped accesses are reported using EC syndrome value 0x03.

TrapEventIDcounter registers. Read Traps EL0 read access to common event identification counters registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2.TGE is 1.

TIDELR	Meaning
0b0	EL0 using AArch32: EL0 reads of the PMXVCNTR and PMEVCNTR<n> , and EL0 read/write accesses to the PMSELR , are trapped if PMUSERENR_EL0.EN is also 0. Accesses EL0 to using PMCEID<n>_EL0 AArch64: and EL0 PMCEID<n> reads are of not trapped by this mechanism. the PMXVCNTR_EL0 and PMEVCNTR<n>_EL0 , and EL0 read/write accesses to the PMSELR_EL0 , are trapped if PMUSERENR_EL0.EN is also 0.
0b1	<ul style="list-style-type: none"> RO access to PMXVCNTR_EL0 and PMEVCNTR<n>_EL0 at EL0. RW access to PMSELR_EL0 at EL0. RW access to PMSELR at EL0. EL0 Overrides read PMUSERENR_EL0.EN accesses to PMCEID<n>_EL0 and PMCEID<n> are trapped. enables:

In AArch64 state, the register accesses affected by this control are:

- MRS reads of [PMCEID0_EL0](#) and [PMCEID1_EL0](#).

In AArch32 state, the register accesses affected by this control are:

- MRC reads of [PMCEID0](#), [PMCEID1](#), [PMCEID2](#), and [PMCEID3](#).

When trapped, reads generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, and:

- AArch64 MRS reads are reported using EC syndrome value 0x18.
- AArch32 MRC reads are reported using EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

CR, bit [2]

In AArch64 state, trapped accesses are reported using EC syndrome value 0x18.

In AArch32 state, trapped MRC accesses are reported using EC syndrome value 0x03, trapped MRRC accesses are reported using EC syndrome value 0x04.

CR	Meaning
0b0	EL0 using AArch64: EL0 read accesses to the PMCCNTR_EL0 are trapped if PMUSERENR_EL0.EN is also 0. EL0 using AArch32: EL0 read accesses to the PMCCNTR are trapped if PMUSERENR_EL0.EN is also 0.
0b1	Overrides PMUSERENR_EL0.EN and enables access to: <ul style="list-style-type: none"> PMCCNTR_EL0 at EL0. PMCCNTR at EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Reserved, RES0.

Cycle counter Read. Traps EL0 access to cycle counter reads to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1.

IRSW, bit [51]

When FEAT_PMuV3_ICNTR is implemented:

Instruction Traps counter Software Read Increment enable. writes to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1.

When PMUSERENR_EL0.UEN is 0, PMUSERENR_EL0.IR enables EL0 reads of the instruction counter.

In AArch64 state, trapped accesses are reported using EC syndrome value 0x18.

When PMUSERENR_EL0.UEN is 1, EL0 reads of the instruction counter and EL0 writes to PMZR_EL0 are enabled by PMUSERENR_EL0.UEN, unless trapped by another control, and PMUSERENR_EL0.IR controls the behavior of EL0 writes to the instruction counter and PMZR_EL0.

In AArch32 state, trapped accesses are reported using EC syndrome value 0x03.

IRSW	Meaning
0b0	When EL0 PMUSERENR_EL0.UEN using == 0 AArch64: EL0 reads/writes to the instruction counter are disabled. PMSWINC_EL0 are trapped if PMUSERENR_EL0.EN is also 0. When EL0 PMUSERENR_EL0.UEN using == 1 AArch32: Permitted EL0 writes are to not affected by this mechanism. the PMSWINC are trapped if PMUSERENR_EL0.EN is also 0.
0b1	When Overrides PMUSERENR_EL0.UEN PMUSERENR_EL0.EN == and 0: enables EL0 access reads of the instruction counter are enabled, unless trapped by another control. to: <ul style="list-style-type: none"> PMSWINC_EL0 at EL0. PMSWINC at EL0. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes to the instruction counter and PMZR_EL0.F0 are ignored.

In AArch64 state, the register accesses affected by this control are:

- When PMUSERENR_EL0.UEN is 0, MRS reads of PMICNTR_EL0.
- When PMUSERENR_EL0.UEN is 1, MSR writes to PMZR_EL0 and PMICNTR_EL0.

When disabled, reads generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and HCR_EL2.TGE is 1, reported using EC syndrome value 0x18.

Ignored writes are not trapped and do not generate an exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UEN, bit [4]

When FEAT_PMUv3p9 is implemented:

User Enable, with access controlled by PMUACR_EL1. Enables EL0 read/write access to PMU registers, other than PMCR_EL0. Software can restrict permitted accesses using PMUSERENR_EL0.{IR,ER,CR} and PMUACR_EL1.

UEN	Meaning
0b0	EL0 accesses to PMU registers are trapped, unless enabled by PMUSERENR_EL0.{IR,ER,CR,EN}.
0b1	EL0 accesses to PMU registers are enabled, unless trapped by another control. A permitted access might be restricted by PMUSERENR_EL0.{IR,ER,CR} and PMUACR_EL1.

In AArch64 state, the register accesses affected by this control are:

- MRS or MSR accesses to the following registers:
 - PMCCFILTR_EL0, PMCCNTR_EL0, PMCNTENCLR_EL0, PMCNTENSET_EL0, PMEVCNTR<n>_EL0, PMEVTYPER<n>_EL0, PMOVSLR_EL0, PMOVSSSET_EL0, PMSELR_EL0, PMXEVCNTR_EL0, and PMXEVTYPER_EL0.
 - If FEAT_PMUv3 ICNTR is implemented, PMICFILTR_EL0 and PMICNTR_EL0.
- MRS reads of PMCEID0_EL0 and PMCEID1_EL0.

- MSR writes to [PMSWINC_EL0](#) and [PMZR_EL0](#).

In AArch32 state, the register accesses affected by this control are:

- MRC or MCR accesses to [PMCCFILTER](#), [PMCCNTR](#), [PMCNTENCLR](#), [PMCNTENSET](#), [PMEVCNTR<n>](#), [PMEVTYPER<n>](#), [PMOVSr](#), [PMOVSSET](#), [PMSELR](#), [PMXEVCNTR](#), and [PMXEVTYPER](#).
- MRC reads of [PMCEID0](#), [PMCEID1](#), [PMCEID2](#), and [PMCEID3](#).
- MCR writes to [PMSWINC](#).
- MRRC or MCRR accesses to [PMCCNTR](#).

When trapped, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, and:

- AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18.
- AArch32 MRC and MCR accesses are reported using EC syndrome value 0x03.
- AArch32 MRRC and MCRR accesses are reported using EC syndrome value 0x04.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ER, bit [3]

When FEAT_PMuV3p9 is implemented:

Event counters Read enable.

When [PMUSERENR_EL0.UEN,EN](#) is {0,0}, [PMUSERENR_EL0.ER](#) enables EL0 reads of the event counters and EL0 reads and writes of the select register.

When [PMUSERENR_EL0.UEN,EN](#) is {1,0}, EL0 reads of the event counters and EL0 writes to [PMZR_EL0](#) are enabled by [PMUSERENR_EL0.UEN](#), unless trapped by another control, and [PMUSERENR_EL0.ER](#) controls the behavior of EL0 writes to the event counters and [PMZR_EL0](#).

ER	Meaning
0b0	When PMUSERENR_EL0.UEN == 0: EL0 reads of the event counters and EL0 reads and writes of the select register are disabled, unless enabled by PMUSERENR_EL0.EN . When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes are not affected by this mechanism.
0b1	When PMUSERENR_EL0.UEN == 0: EL0 reads of the event counters and EL0 reads and writes of the select register are enabled, unless trapped by another control. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes to the event counters and PMZR_EL0.P[30:0] are ignored, unless enabled by PMUSERENR_EL0.EN .

In AArch64 state, the register accesses affected by this control are:

- When [PMUSERENR_EL0.UEN,EN](#) is {0,0}:
 - MRS reads of [PMEVCNTR<n>_EL0](#) and [PMXEVCNTR_EL0](#).
 - MRS and MSR accesses to [PMSELR_EL0](#).
- When [PMUSERENR_EL0.UEN,EN](#) is {1,0}, MSR writes to [PMZR_EL0](#), [PMEVCNTR<n>_EL0](#) and [PMXEVCNTR_EL0](#).

In AArch32 state, the register accesses affected by this control are:

- When [PMUSERENR_EL0.UEN,EN](#) is {0,0}:
 - MRC reads of [PMEVCNTR<n>](#) and [PMXEVCNTR](#).
 - MRC and MCR accesses to [PMSELR](#).
- When [PMUSERENR_EL0.UEN,EN](#) is {1,0}, MCR writes to [PMEVCNTR<n>](#) and [PMXEVCNTR](#).

When disabled, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, and:

- AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18.
- AArch32 MRC and MCR accesses are reported using EC syndrome value 0x03.

Ignored writes are not trapped and do not generate an exception.

This field is ignored by the PE when `PMUSERENR_EL0.EN == 1`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Event counters Read enable.

When `PMUSERENR_EL0.EN` is 0, `PMUSERENR_EL0.ER` enables EL0 reads of the event counters and EL0 reads and writes of the select register.

ER	Meaning
0b0	EL0 reads of the event counters and EL0 reads and writes of the select register are disabled, unless enabled by <code>PMUSERENR_EL0.EN</code> .
0b1	EL0 reads of the event counters and EL0 reads and writes of the select register are enabled, unless trapped by another control.

In AArch64 state, the register accesses affected by this control are:

- MRS reads of `PMEVCNTR<n>_EL0` and `PMXVCNTR_EL0`.
- MRS and MSR accesses to `PMSELR_EL0`.

In AArch32 state, the register accesses affected by this control are:

- MRC reads of `PMEVCNTR<n>` and `PMXVCNTR`.
- MRC and MCR accesses to `PMSELR`.

When disabled, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and `HCR_EL2.TGE` is 1, and:

- AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18.
- AArch32 MRC and MCR accesses are reported using EC syndrome value 0x03.

This field is ignored by the PE when `PMUSERENR_EL0.EN == 1`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CR, bit [2]

When FEAT_PMuV3p9 is implemented:

Cycle counter Read enable.

When `PMUSERENR_EL0.{UEN,EN}` is {0,0}, `PMUSERENR_EL0.CR` enables EL0 reads of the cycle counter.

When `PMUSERENR_EL0.{UEN,EN}` is {1,0}, EL0 reads of the cycle counter and EL0 writes to `PMZR_EL0` are enabled by `PMUSERENR_EL0.UEN`, unless trapped by another control, and `PMUSERENR_EL0.CR` controls the behavior of EL0 writes to the cycle counter and `PMZR_EL0`.

CR	Meaning
0b0	When PMUSERENR_EL0.UEN == 0: EL0 reads of the cycle counter are disabled, unless enabled by PMUSERENR_EL0.EN. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes are not affected by this mechanism.
0b1	When PMUSERENR_EL0.UEN == 0: EL0 reads of the cycle counter are enabled, unless trapped by another control. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes to the cycle counter and PMZR_EL0.C are ignored, unless enabled by PMUSERENR_EL0.EN.

In AArch64 state, the register accesses affected by this control are:

- When PMUSERENR_EL0.{UEN,EN} is {0,0}, MRS reads of PMCCNTR_EL0.
- When PMUSERENR_EL0.{UEN,EN} is {1,0}, MSR writes to PMZR_EL0 and PMCCNTR_EL0.

In AArch32 state, the register accesses affected by this control are:

- When PMUSERENR_EL0.{UEN,EN} is {0,0}:
 - MRC reads of PMCCNTR.
 - MRRC reads of PMCCNTR.
- When PMUSERENR_EL0.{UEN,EN} is {1,0}:
 - MCR writes to PMCCNTR.
 - MCRR writes to PMCCNTR.

When disabled, reads generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and HCR_EL2.TGE is 1, and:

- AArch64 MRS reads are reported using EC syndrome value 0x18.
- AArch32 MRC reads are reported using EC syndrome value 0x03.
- AArch32 MRRC reads are reported using EC syndrome value 0x04.

Ignored writes are not trapped and do not generate an exception.

This field is ignored by the PE when PMUSERENR_EL0.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Cycle counter Read enable.

When PMUSERENR_EL0.EN is 0, PMUSERENR_EL0.CR enables EL0 reads of the cycle counter.

CR	Meaning
0b0	EL0 reads of the cycle counter are disabled, unless enabled by PMUSERENR_EL0.EN.
0b1	EL0 reads of the cycle counter are enabled, unless trapped by another control.

In AArch64 state, the register accesses affected by this control are:

- MRS reads of PMCCNTR_EL0.

In AArch32 state, the register accesses affected by this control are:

- MRC reads of PMCCNTR.
- MRRC reads of PMCCNTR.

When disabled, reads generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and HCR_EL2.TGE is 1, and:

- AArch64 MRS reads are reported using EC syndrome value 0x18.
- AArch32 MRC reads are reported using EC syndrome value 0x03.
- AArch32 MRRC reads are reported using EC syndrome value 0x04.

This field is ignored by the PE when PMUSERENR_EL0.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SW, bit [1]

When FEAT_PMuV3p9 is implemented:

Software increment register Write enable.

When PMUSERENR_EL0.UEN is 0, PMUSERENR_EL0.SW enables EL0 writes to the Software increment register.

When PMUSERENR_EL0.UEN is 1, EL0 writes to the Software increment register are enabled by PMUSERENR_EL0.UEN, unless trapped by another control, and PMUSERENR_EL0.SW controls the behavior of EL0 writes to the Software increment register.

SW	Meaning
0b0	When PMUSERENR_EL0.UEN == 0: EL0 writes to the Software increment register are disabled, unless enabled by PMUSERENR_EL0.EN. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes are not affected by this mechanism.
0b1	When PMUSERENR_EL0.UEN == 0: EL0 writes to the Software increment register are enabled, unless trapped by another control. When PMUSERENR_EL0.UEN == 1: Permitted EL0 writes to the Software increment register ignore the value of PMUACR_EL1.

In AArch64 state, the register accesses affected by this control are:

- MSR writes to PMSWINC_EL0.

In AArch32 state, the register accesses affected by this control are:

- MCR writes to PMSWINC.

When disabled, writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and HCR_EL2.TGE is 1, and:

- AArch64 MSR writes are reported using EC syndrome value 0x18.
- AArch32 MCR writes are reported using EC syndrome value 0x03.

This field is ignored by the PE when PMUSERENR_EL0.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Software increment register Write enable.

When PMUSERENR_EL0.EN is 0, PMUSERENR_EL0.SW enables EL0 writes to the Software increment register.

SW	Meaning
0b0	EL0 writes to the Software increment register are disabled, unless enabled by PMUSERENR_EL0.EN.
0b1	EL0 writes to the Software increment register are enabled, unless trapped by another control.

In AArch64 state, the register accesses affected by this control are:

- MSR writes to PMSWINC_EL0.

In AArch32 state, the register accesses affected by this control are:

- MCR writes to PMSWINC.

When disabled, writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, and:

- AArch64 MSR writes are reported using EC syndrome value 0x18.
- AArch32 MCR writes are reported using EC syndrome value 0x03.

This field is ignored by the PE when `PMUSERENR_EL0.EN == 1`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EN, bit [0]

- In AArch64 state, MRS or MSR accesses to the following registers are reported using EC syndrome value 0x18:
 - [PMCR_EL0](#), [PMOVSCLR_EL0](#), [PMSELR_EL0](#), [PMCEID0_EL0](#), [PMCEID1_EL0](#), [PMCCNTR_EL0](#), [PMXEVTYPER_EL0](#), [PMXVCNTR_EL0](#), [PMCNTENSET_EL0](#), [PMCNTENCLR_EL0](#), [PMOVSSET_EL0](#), [PMEVCNTR<n>_EL0](#), [PMEVTYPER<n>_EL0](#), [PMCCFILTR_EL0](#), [PMSWINC_EL0](#).
 - If FEAT_PMUv3p4 is implemented, [PMMIR_EL1](#).
- In AArch32 state, MRC or MCR accesses to the following registers are reported using EC syndrome value 0x03:
 - [PMCR](#), [PMOVS](#), [PMSEL](#), [PMCEID0](#), [PMCEID1](#), [PMCCNTR](#), [PMXEVTYPER](#), [PMXVCNTR](#), [PMCNTENSET](#), [PMCNTENCLR](#), [PMOVSSET](#), [PMEVCNTR<n>](#), [PMEVTYPER<n>](#), [PMCCFILTR](#), [PMSWINC](#).
 - If FEAT_PMUv3p4 is implemented, [PMMIR](#).
 - If FEAT_PMUv3p1 is implemented, in AArch32 state, [PMCEID2](#), and [PMCEID3](#).
- In AArch32 state, MRRC or MCRR accesses to [PMCCNTR](#) are reported using EC syndrome value 0x04.

Enable. Traps Enables EL0 read/write accesses access to PMU the Performance Monitor registers to EL1, other or than to EL2 when it is implemented and enabled for the instruction current counter. Security state and [HCR_EL2.TGE](#) is 1, from both Execution states as follows:

EN	Meaning
0b0	While at EL0, accesses to PMU the specified registers at EL0 are trapped, unless enabled overridden by one of <code>PMUSERENR_EL0.{UEN, ER, CR, SW}</code> .
0b1	EL0 While accesses at to EL0, PMU software registers can are access enabled, all unless of trapped the by specified another control registers.

In AArch64 state, the register accesses affected by this control are:

- MRS or MSR accesses to [PMCCFILTR_EL0](#), [PMCCNTR_EL0](#), [PMCNTENCLR_EL0](#), [PMCNTENSET_EL0](#), [PMCR_EL0](#), [PMEVCNTR<n>_EL0](#), [PMEVTYPER<n>_EL0](#), [PMOVSCLR_EL0](#), [PMOVSSET_EL0](#), [PMSELR_EL0](#), [PMXVCNTR_EL0](#), and [PMXEVTYPER_EL0](#).
- MRS reads of [PMCEID0_EL0](#) and [PMCEID1_EL0](#).
- MSR writes to the following registers:
 - [PMSWINC_EL0](#).
 - If FEAT_PMUv3p9 is implemented, [PMZR_EL0](#).

Note

When FEAT_PMUv3_ICNTR is implemented, this field does not affect MRS and MSR accesses to [PMICNTR_EL0](#) and [PMICFILTR_EL0](#).

In AArch32 state, the register accesses affected by this control are:

- MRC or MCR accesses to [PMCCFILTR](#), [PMCCNTR](#), [PMCNTENCLR](#), [PMCNTENSET](#), [PMCR](#), [PMEVCNTR<n>](#), [PMEVTYPER<n>](#), [PMOVS](#), [PMOVSSET](#), [PMSEL](#), [PMXVCNTR](#), and [PMXEVTYPER](#).
- MRC reads of the following registers:

- [PMCEID0](#) and [PMCEID1](#).
- If FEAT_PMUv3p1 is implemented, [PMCEID2](#) and [PMCEID3](#).
- MCR writes to [PMSWINC](#).
- MRRC or MCRR accesses to [PMCCNTR](#).

When trapped, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, and:

- AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18.
- AArch32 MRC and MCR accesses are reported using EC syndrome value 0x03.
- AArch32 MRRC and MCRR accesses are reported using EC syndrome value 0x04.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMUSERENR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMUSERENR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMUSERENR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMUSERENR_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMUSERENR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMUSERENR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMUSERENR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMUSERENR_EL0;

```

MSR PMUSERENR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMUSERENR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMUSERENR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMUSERENR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMUSERENR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMXEVCNTR_EL0, Performance Monitors Selected Event Count Register

The PMXEVCNTR_EL0 characteristics are:

Purpose

Reads or writes the value of the selected event counter, [PMEVCNTR<n>_EL0](#). [PMSELR_EL0](#).SEL determines which event counter is selected.

Configuration

AArch64 System register PMXEVCNTR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMEVCNTR\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMXEVCNTR_EL0 are UNDEFINED.

Attributes

PMXEVCNTR_EL0 is a 64-bit register.

Field descriptions

When FEAT_PMUv3p5 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
PMEVCNTR<n>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMEVCNTR<n>																															

PMEVCNTR<n>, bits [63:0]

Value of the selected event counter, [PMEVCNTR<n>_EL0](#), where n is the value stored in [PMSELR_EL0](#).SEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
PMEVCNTR<n>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

PMEVCNTR<n>, bits [31:0]

Value of the selected event counter, [PMEVCNTR<n>_EL0](#), where n is the value stored in [PMSELR_EL0](#).SEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMXEVCNTR_EL0

If FEAT_FGT is implemented and [PMSELR_EL0.SEL](#) is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMXEVCNTR_EL0](#) is as follows:

- If [PMSELR_EL0.SEL](#) selects an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and [PMSELR_EL0.SEL](#) is greater than or equal to the number of accessible event counters, then reads and writes of [PMXEVCNTR_EL0](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP
- Accesses to the register behave as if [PMSELR_EL0.SEL](#) has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.
- If EL2 is implemented and enabled in the current Security state, and [PMSELR_EL0.SEL](#) is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR_EL0](#).{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, [MDCR_EL2](#).HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [MDCR_EL2](#).HPMN.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMXEVCNTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b010

```

if UInt(PMSELR_EL0.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && UInt(PMSELR_EL0.SEL) >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && UInt(PMSELR_EL0.SEL) >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)];

```

MSR PMXEVCNTR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b010

```

if UInt(PMSELR_EL0.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && UInt(PMSELR_EL0.SEL) >= AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && UInt(PMSELR_EL0.SEL) >= AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMXEVTYPER_ELO, Performance Monitors Selected Event Type Register

The PMXEVTYPER_ELO characteristics are:

Purpose

When [PMSELR_ELO](#).SEL selects an event counter, this accesses a [PMEVTYPER<n>_ELO](#) register. When [PMSELR_ELO](#).SEL selects the cycle counter, this accesses [PMCCFILTR_ELO](#).

Configuration

AArch64 System register PMXEVTYPER_ELO bits [31:0] are architecturally mapped to AArch32 System register [PMXEVTYPER\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMXEVTYPER_ELO are UNDEFINED.

Attributes

PMXEVTYPER_ELO is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Event type register or PMCCFILTR_ELO																															
Event type register or PMCCFILTR_ELO																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

When [PMSELR_ELO](#).SEL == 31, this register accesses [PMCCFILTR_ELO](#).

Otherwise, this register accesses [PMEVTYPER<n>_ELO](#) where n is the value in [PMSELR_ELO](#).SEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMXEVTYPER_ELO

If FEAT_FGT is implemented, and [PMSELR_ELO](#).SEL is not 31 and is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMXEVTYPER_ELO](#) is as follows:

- If [PMSELR_ELO](#).SEL selects an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented, and [PMSELR_ELO](#).SEL is not 31 and is greater than or equal to the number of accessible event counters, then reads and writes of [PMXEVTYPER_ELO](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR_ELO](#).SEL has an UNKNOWN value less than the number of event counters accessible at the current Exception level and Security state.

- Accesses to the register behave as if [PMSELR_EL0.SEL](#) is 31.
- If EL2 is implemented and enabled in the current Security state, [PMSELR_EL0](#) is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR_EL0.EN](#).

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, [MDCR_EL2.HPMN](#) identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [MDCR_EL2.HPMN](#).

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, PMXEVTYPER_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b001


```

if UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >=
AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif UInt(PMSCLR_EL0.SEL) == 31 then
            X[t, 64] = PMCCFILTR_EL0;
        else
            X[t, 64] = PMEVTYPER_EL0[UInt(PMSCLR_EL0.SEL)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif UInt(PMSCLR_EL0.SEL) == 31 then
        X[t, 64] = PMCCFILTR_EL0;
    else
        X[t, 64] = PMEVTYPER_EL0[UInt(PMSCLR_EL0.SEL)];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif UInt(PMSCLR_EL0.SEL) == 31 then
        X[t, 64] = PMCCFILTR_EL0;
    else

```

```

        X[t, 64] = PMEVTYPER_EL0[UInt(PMSELR_EL0.SEL)];
elsif PSTATE.EL == EL3 then
    if UInt(PMSELR_EL0.SEL) == 31 then
        X[t, 64] = PMCCFILTR_EL0;
    else
        X[t, 64] = PMEVTYPER_EL0[UInt(PMSELR_EL0.SEL)];

```

MSR PMXEVTYPER_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b001

```

if UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVTYPEPn_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >=
AArch64.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            else
                AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif UInt(PMSCLR_EL0.SEL) == 31 then
            PMCCFILTR_EL0 = X[t, 64];
        else
            PMEVTYPERS_EL0[UInt(PMSCLR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.PMEVTYPEPn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && UInt(PMSCLR_EL0.SEL) != 31 && UInt(PMSCLR_EL0.SEL) >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif UInt(PMSCLR_EL0.SEL) == 31 then
        PMCCFILTR_EL0 = X[t, 64];
    else
        PMEVTYPERS_EL0[UInt(PMSCLR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif UInt(PMSCLR_EL0.SEL) == 31 then
        PMCCFILTR_EL0 = X[t, 64];
    else

```

```

    PMEVTYPER_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if UInt(PMSELR_EL0.SEL) == 31 then
        PMCCFILTR_EL0 = X[t, 64];
    else
        PMEVTYPER_EL0[UInt(PMSELR_EL0.SEL)] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMZR_EL0, Performance Monitors Zero with Mask

The PMZR_EL0 characteristics are:

Purpose

Zero the set of counters specified by the mask written to PMZR_EL0.

Configuration

This register is present only when FEAT_PMUv3p9 is implemented. Otherwise, direct accesses to PMZR_EL0 are UNDEFINED.

Attributes

PMZR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

F<m>, bit [m+32], for m = 0
When FEAT_PMUv3_ICNTR is implemented:

Zero fixed-function instruction counter.

F<m>	Meaning
0b0	Do not set fixed-function counter <m> to zero.
0b1	Set fixed-function counter <m> to zero.

Writing 1 to PMZR_EL0.F[0] zeros the instruction counter, [PMICNTR_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

Zero [PMCCNTR_EL0](#).

C	Meaning
0b0	Do not set Cycle counter to zero.
0b1	Set Cycle counter to zero.

Access to this field is **W1C**.

P<m>, bit [m], for m = 30 to 0

Zero [PMEVCNTR<n>_EL0](#).

P<m>	Meaning
0b0	Do not set Event counter <m> to zero.
0b1	Set Event counter <m> to zero.

Access to this field is **W1C**.

Accessing PMZR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MSR PMZR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMZR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMZR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMZR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMZR_EL0 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

POR_EL0, Permission Overlay Register 0 (EL0)

The POR_EL0 characteristics are:

Purpose

Stage 1 Permission Overlay Register for unprivileged access of EL1&0 or EL2&0 translation regime.

Configuration

This register is present only when FEAT_S1POE is implemented. Otherwise, direct accesses to POR_EL0 are UNDEFINED.

Attributes

POR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Perm Represents Stage 1 Overlay Permissions.

Perm<m>	Meaning
0b0000	No access.
0b0001	Read.
0b0010	Execute.
0b0011	Read, Execute.
0b0100	Write.
0b0101	Write, Read.
0b0110	Write, Execute.
0b0111	Read, Write, Execute.
0b1xxx	Reserved - treated as No access

When VMSAv9-128 is not in use, fields Perm[8] to Perm[15] are not used.

This field is not permitted to be cached in a TLB.

When Stage 1 Overlay mechanism is disabled, contents of this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing POR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, POR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elseif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.E0POE == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.nPOR_EL0 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.E0POE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = POR_EL0;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elseif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPOR_EL0 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = POR_EL0;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elseif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = POR_EL0;
elseif PSTATE.EL == EL3 then
    X[t, 64] = POR_EL0;

```

MSR POR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.E0POE == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TVM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.nPOR_EL0 == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.E0POE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                POR_EL0 = X[t, 64];
        elsif PSTATE.EL == EL1 then
            if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
                UNDEFINED;
            elsif EL2Enabled() && HCR_EL2.TVM == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPOR_EL0 == '0' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.SystemAccessTrap(EL3, 0x18);
                else
                    POR_EL0 = X[t, 64];
            elsif PSTATE.EL == EL2 then
                if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
                    UNDEFINED;
                elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
                    if Halted() && EDSCR.SDD == '1' then
                        UNDEFINED;
                    else
                        AArch64.SystemAccessTrap(EL3, 0x18);
                    else
                        POR_EL0 = X[t, 64];
            elsif PSTATE.EL == EL3 then
                POR_EL0 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

POR_EL1, Permission Overlay Register 1 (EL1)

The POR_EL1 characteristics are:

Purpose

Stage 1 Permission Overlay Register for privileged access of the EL1&0 translation regime.

Configuration

This register is present only when FEAT_S1POE is implemented. Otherwise, direct accesses to POR_EL1 are UNDEFINED.

Attributes

POR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Perm Represents Stage 1 Overlay Permissions.

Perm<m>	Meaning
0b0000	No access.
0b0001	Read.
0b0010	Execute.
0b0011	Read, Execute.
0b0100	Write.
0b0101	Write, Read.
0b0110	Write, Execute.
0b0111	Read, Write, Execute.
0b1xxx	Reserved - treated as No access

When VMSAv9-128 is not in use, fields Perm[8] to Perm[15] are not used.

This field is not permitted to be cached in a TLB.

When Stage 1 Overlay mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing POR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, POR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPOR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x2A8];
    else
        X[t, 64] = POR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = POR_EL2;
    else
        X[t, 64] = POR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = POR_EL1;

```

MSR POR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPOR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x2A8] = X[t, 64];
    else
        POR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        POR_EL2 = X[t, 64];
    else
        POR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    POR_EL1 = X[t, 64];

```

MRS <Xt>, POR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x2A8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = POR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = POR_EL1;
    else
        UNDEFINED;

```

MSR POR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x2A8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            POR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        POR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

POR_EL2, Permission Overlay Register 2 (EL2)

The POR_EL2 characteristics are:

Purpose

Stage 1 Permission Overlay Register for privileged access of the EL2 or EL2&0 translation regime.

Configuration

This register is present only when FEAT_S1POE is implemented. Otherwise, direct accesses to POR_EL2 are UNDEFINED.

Attributes

POR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8																								
Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Perm Represents Stage 1 Overlay Permissions.

Perm<m>	Meaning
0b0000	No access.
0b0001	Read.
0b0010	Execute.
0b0011	Read, Execute.
0b0100	Write.
0b0101	Write, Read.
0b0110	Write, Execute.
0b0111	Read, Write, Execute.
0b1xxx	Reserved - treated as No access

When VMSAv9-128 is not in use, fields Perm[8] to Perm[15] are not used.

This field is not permitted to be cached in a TLB.

When Stage 1 Overlay mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing POR_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name POR_EL2 or [POR_EL1](#) are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, POR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = POR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = POR_EL2;

```

MSR POR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        POR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    POR_EL2 = X[t, 64];

```

MRS <Xt>, POR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nPOR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x2A8];
    else
        X[t, 64] = POR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = POR_EL2;
    else
        X[t, 64] = POR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = POR_EL1;

```

MSR POR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nPOR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x2A8] = X[t, 64];
    else
        POR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        POR_EL2 = X[t, 64];
    else
        POR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    POR_EL1 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

POR_EL3, Permission Overlay Register 3 (EL3)

The POR_EL3 characteristics are:

Purpose

Stage 1 Permission Overlay Register for privileged access of the EL3 translation regime.

Configuration

This register is present only when FEAT_S1POE is implemented. Otherwise, direct accesses to POR_EL3 are UNDEFINED.

Attributes

POR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Perm Represents Stage 1 Overlay Permissions.

Perm<m>	Meaning
0b0000	No access.
0b0001	Read.
0b0010	Execute.
0b0011	Read, Execute.
0b0100	Write.
0b0101	Write, Read.
0b0110	Write, Execute.
0b0111	Read, Write, Execute.
0b1xxx	Reserved - treated as No access

When VMSAv9-128 is not in use, fields Perm[8] to Perm[15] are not used.

This field is not permitted to be cached in a TLB.

When Stage 1 Overlay mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing POR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, POR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = POR_EL3;

```

MSR POR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    POR_EL3 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

RCWMASK_EL1, Read Check Write Instruction Mask (EL1)

The RCWMASK_EL1 characteristics are:

Purpose

Contains the mask used by RCW instructions.

Configuration

This register is present only when FEAT_THE is implemented. Otherwise, direct accesses to RCWMASK_EL1 are UNDEFINED.

RCWMASK_EL1 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Attributes

RCWMASK_EL1 is a:

- 128-bit register when FEAT_D128 is implemented
- 64-bit register otherwise

Field descriptions

When FEAT_D128 is implemented:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98
Mask[38]	RES0	Mask[37:34]	RES0	Mask[33:30]	RES0	Mask[29]	RES0	Mask[28:26]	Mask				RES0																Mask
	Mask[25:16]																												

Mask, bits [127, 124:121, 118:115, 113, 111:109, 100:91, 16:1]

Mask used to decide which bit-fields are writable to the 128-bit Descriptor by RCW or RCWS Instructions.

The Mask field is split as follows:

- Mask[38] is RCWMASK_EL1[127].
- Mask[37:34] is RCWMASK_EL1[124:121].
- Mask[33:30] is RCWMASK_EL1[118:115].
- Mask[29] is RCWMASK_EL1[113].
- Mask[28:26] is RCWMASK_EL1[111:109].
- Mask[25:16] is RCWMASK_EL1[100:91].
- Mask[15:0] is RCWMASK_EL1[16:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [126:125]

Reserved, RES0.

Bits [120:119]

Reserved, RES0.

Bit [114]

Reserved, RES0.

Bit [112]

Reserved, RES0.

Mask, bit [108]**When FEAT_MEC is implemented:**

Mask used to decide which bit-fields are writable to the 128-Bit Descriptor by RCW or RCWS Instructions, aligning with the 128-bit AMEC format.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [107:101]

Reserved, RES0.

Bits [90:17]

Reserved, RES0.

Bit [0]

Reserved, RES0.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
Mask[29:19]											RES0	Mask[18:17]											RES0												
RES0														Mask[16:0]																		RES0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Mask, bits [63:53, 51:50, 17:1]

Mask used to decide which bit-fields are writable to the 64-bit Descriptor by RCW or RCWS Instructions.

The Mask field is split as follows:

- Mask[29:19] is RCWMASK_EL1[63:53].
- Mask[18:17] is RCWMASK_EL1[51:50].
- Mask[16:0] is RCWMASK_EL1[17:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [52]

Reserved, RES0.

Bits [49:18]

Reserved, RES0.

Bit [0]

Reserved, RES0.

Accessing RCWMASK_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, RCWMASK_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nRCWMASK_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = RCWMASK_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = RCWMASK_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = RCWMASK_EL1<63:0>;

```

MSR RCWMASK_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b110


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nRCWMASK_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            RCWMASK_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                RCWMASK_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL3 then
        RCWMASK_EL1<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, RCWMASK_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nRCWMASK_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>);
    elsif PSTATE.EL == EL3 then
        (X[t + 1, 64], X[t, 64]) = (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR RCWMASK_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nRCWMASK_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (RCWMASK_EL1<127:64>, RCWMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

RCWSMASK_EL1, Software Read Check Write Instruction Mask (EL1)

The RCWSMASK_EL1 characteristics are:

Purpose

Contains the software mask used by RCWS instructions.

Configuration

This register is present only when FEAT_THE is implemented. Otherwise, direct accesses to RCWSMASK_EL1 are UNDEFINED.

RCWSMASK_EL1 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Attributes

RCWSMASK_EL1 is a:

- 128-bit register when FEAT_D128 is implemented
- 64-bit register otherwise

Field descriptions

When FEAT_D128 is implemented:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111
Software_Mask[38]	RES0	Software_Mask[37:34]	RES0	Software_Mask[33:30]	RES0	Software_Mask[29]	RES0	Software_Mask[28:26]	RES0	Software_Mask[25:16]	RES0	Software_Mask[15:0]	RES0	Software_Mask[14:12]	RES0	Software_Mask[11:9]
RES0																RES0
RES0																RES0

Software_Mask, bits [127, 124:121, 118:115, 113, 111:109, 100:91, 16:1]

Software Mask used to decide which bit-fields are writable to the 128-bit Descriptor by RCWS Instruction.

The Software_Mask field is split as follows:

- Software_Mask[38] is RCWSMASK_EL1[127].
- Software_Mask[37:34] is RCWSMASK_EL1[124:121].
- Software_Mask[33:30] is RCWSMASK_EL1[118:115].
- Software_Mask[29] is RCWSMASK_EL1[113].
- Software_Mask[28:26] is RCWSMASK_EL1[111:109].
- Software_Mask[25:16] is RCWSMASK_EL1[100:91].
- Software_Mask[15:0] is RCWSMASK_EL1[16:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [126:125]

Reserved, RES0.

Bits [120:119]

Reserved, RES0.

Bit [114]

Reserved, RES0.

Bit [112]

Reserved, RES0.

Mask, bit [108]**When FEAT_THE is implemented and FEAT_MEC is implemented:**

Mask used to decide which bit-fields are writable to the 128-Bit Descriptor by RCWS Instructions, aligning with the 128-bit AMEC format.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [107:101]

Reserved, RES0.

Bits [90:17]

Reserved, RES0.

Bit [0]

Reserved, RES0.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
Software_Mask[30:17]														RES0																		
RES0														Software_Mask[16:0]																	RES0	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Software_Mask, bits [63:50, 17:1]

Software Mask used to decide which bit-fields are writable to the 64-bit Descriptor by RCWS Instruction.

The Software_Mask field is split as follows:

- Software_Mask[30:17] is RCWSMASK_EL1[63:50].
- Software_Mask[16:0] is RCWSMASK_EL1[17:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [49:18]

Reserved, RES0.

Bit [0]

Reserved, RES0.

Accessing RCWSMASK_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, RCWSMASK_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGTR2_EL2.nRCWSMASK_EL1 ==
'0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = RCWSMASK_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = RCWSMASK_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = RCWSMASK_EL1<63:0>;

```

MSR RCWSMASK_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGWTR2_EL2.nRCWSMASK_EL1 ==
'0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            RCWSMASK_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            RCWSMASK_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL3 then
        RCWSMASK_EL1<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, RCWSMASK_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGTR2_EL2.nRCWSMASK_EL1 ==
'0' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (X[t + 1, 64], X[t, 64]) = (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKEn == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (X[t + 1, 64], X[t, 64]) = (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR RCWSMASK_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn2 == '1') && HFGWTR2_EL2.nRCWSMASK_EL1 ==
'0' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.RCWMASKE n == '0' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.RCWMASKE n == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (RCWSMASK_EL1<127:64>, RCWSMASK_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

REVIDR_EL1, Revision ID Register

The REVIDR_EL1 characteristics are:

Purpose

Provides implementation-specific minor revision information.

Configuration

AArch64 System register REVIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [REVIDR\[31:0\]](#).

If REVIDR_EL1 has the same value as [MIDR_EL1](#), then its contents have no significance.

Attributes

REVIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing REVIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, REVIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b110

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
        && HFGTR_EL2.REVIDR_EL1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = REVIDR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = REVIDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = REVIDR_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

RGSR_EL1, Random Allocation Tag Seed Register.

The RGSR_EL1 characteristics are:

Purpose

Random Allocation Tag Seed Register.

Configuration

This register is present only when FEAT_MTE2 is implemented. Otherwise, direct accesses to RGSR_EL1 are UNDEFINED.

When [GCR_EL1.RRND](#)==0b1, updates to RGSR_EL1 are implementation-specific.

When [GCR_EL1.RRND](#)==0b0, direct and indirect reads and writes to the register appear to occur in program order relative to other instructions, without the need for any explicit synchronization.

Attributes

RGSR_EL1 is a 64-bit register.

Field descriptions

When GCR_EL1.RRND == 0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
RES0								SEED														RES0				TAG					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:24]

Reserved, RES0.

SEED, bits [23:8]

Seed register used for generating values returned by RandomAllocationTag().

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:4]

Reserved, RES0.

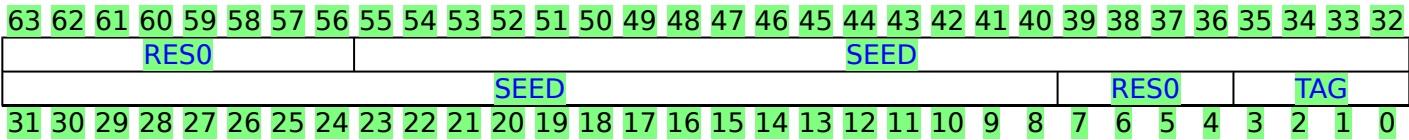
TAG, bits [3:0]

Tag generated by the most recent IRG instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:



Bits [63:56]

Reserved, RES0.

SEED, bits [55:8]

IMPLEMENTATION DEFINED.

Note

Software is recommended to avoid writing SEED[15:0] with a value of zero, unless this has been generated by the PE in response to an earlier value with SEED being nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:4]

Reserved, RES0.

TAG, bits [3:0]

Tag generated by the most recent IRG instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing RGSR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, RGSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ATA == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.ATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.ATA == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = RGSR_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ATA == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.ATA == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = RGSR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = RGSR_EL1;

```

MSR RGSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ATA == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.ATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.ATA == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            RGSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.ATA == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.ATA == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            RGSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        RGSR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)	htmldiff from-	(new)
-------	----------------	-------

SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL S1_<op1>_<Cn>_<Cm>_<op2>, SYSP S1_<op1>_<Cn>_<Cm>_<op2>, IMPLEMENTATION DEFINED maintenance instructions

The SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL S1_<op1>_<Cn>_<Cm>_<op2>, SYSP S1_<op1>_<Cn>_<Cm>_<op2> characteristics are:

Purpose

This area of the System instruction encoding space is reserved for IMPLEMENTATION DEFINED System instructions.

Configuration

There are no configuration notes.

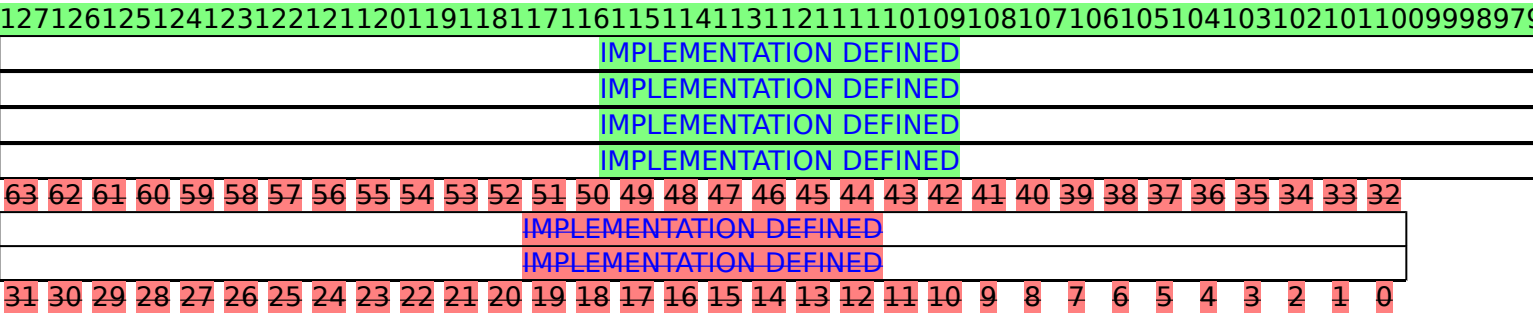
Attributes

SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL S1_<op1>_<Cn>_<Cm>_<op2>, SYSP S1_<op1>_<Cn>_<Cm>_<op2> is a: 64-bit System instruction.

- 128-bit System instruction when FEAT_SYSINSTR128 is implemented
- 64-bit System instruction otherwise

Field descriptions

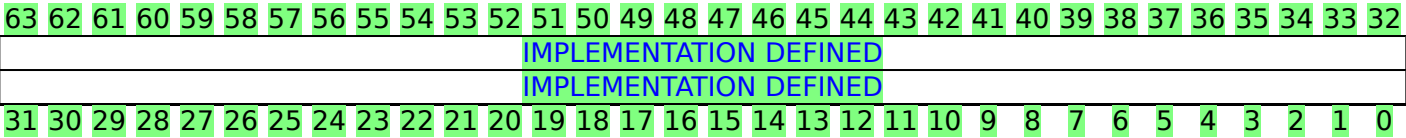
When FEAT_SYSINSTR128 is implemented:



IMPLEMENTATION DEFINED, bits [127:0]

IMPLEMENTATION DEFINED.

Otherwise:



SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL S1_<op1>_<Cn>_<Cm>_<op2>, SYSP S1_<op1>_<Cn>_<Cm>_<op2>, IMPLEMENTATION DEFINED maintenance instructions

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Executing the SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL S1_<op1>_<Cn>_<Cm>_<op2>, SYSP S1_<op1>_<Cn>_<Cm>_<op2> instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

SYS #<op1>, <Cn>, <Cm>, #<op2>{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysInstr(1, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysInstr(1, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL2 then
        AArch64.ImpDefSysInstr(1, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL3 then
        AArch64.ImpDefSysInstr(1, op1, CRn, CRm, op2, t);

```

SYSL <Xt>, #<op1>, <Cn>, <Cm>, #<op2>

op0	op1	CRn	CRm	op2
0b01	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysInstrWithResult(1, op1, CRn, CRm, op2);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysInstrWithResult(1, op1, CRn, CRm, op2);
    elsif PSTATE.EL == EL2 then
        AArch64.ImpDefSysInstrWithResult(1, op1, CRn, CRm, op2);
    elsif PSTATE.EL == EL3 then
        AArch64.ImpDefSysInstrWithResult(1, op1, CRn, CRm, op2);

```


S2PIR_EL2, Stage 2 Permission Indirection Register (EL2)

The S2PIR_EL2 characteristics are:

Purpose

Stage 2 Permission Indirection Register for EL1&0 translation regime.

Configuration

This register is present only when FEAT_S2PIE is implemented. Otherwise, direct accesses to S2PIR_EL2 are UNDEFINED.

Attributes

S2PIR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Represents stage 2 Base Permissions.

Perm<m>	Meaning
0b0000	No Access.
0b0001	Reserved - treated as No Access.
0b0010	MRO.
0b0011	MRO-TL1.
0b0100	WO.
0b0101	Reserved - treated as No Access.
0b0110	MRO-TL0.
0b0111	MRO-TL01.
0b1000	RO.
0b1001	RO+uX.
0b1010	RO+pX.
0b1011	RO+puX.
0b1100	RW.
0b1101	RW+uX.
0b1110	RW+pX.
0b1111	RW+puX.

This field is permitted to be cached in a TLB.

When stage 2 Indirect Permission mechanism is disabled, the contents of this register are ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing S2PIR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, S2PIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x2B0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = S2PIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = S2PIR_EL2;

```

MSR S2PIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x2B0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        S2PIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    S2PIR_EL2 = X[t, 64];

```

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

S2POR_EL1, Stage 2 Permission Overlay Register (EL1)

The S2POR_EL1 characteristics are:

Purpose

Stage 2 Permission Overlay Register for EL1&0 translation regime.

Configuration

This register is present only when FEAT_S2POE is implemented. Otherwise, direct accesses to S2POR_EL1 are UNDEFINED.

Attributes

S2POR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Perm15	Perm14	Perm13	Perm12	Perm11	Perm10	Perm9	Perm8	Perm7	Perm6	Perm5	Perm4	Perm3	Perm2	Perm1	Perm0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Perm<m>, bits [4m+3:4m], for m = 15 to 0

Configures stage 2 Overlay Permissions.

Perm<m>	Meaning
0b0000	No Access.
0b0001	Reserved - treated as No Access.
0b0010	MRO.
0b0011	MRO-TL1.
0b0100	WO.
0b0101	Reserved - treated as No Access.
0b0110	MRO-TL0.
0b0111	MRO-TL01.
0b1000	RO.
0b1001	RO+uX.
0b1010	RO+pX.
0b1011	RO+puX.
0b1100	RW.
0b1101	RW+uX.
0b1110	RW+pX.
0b1111	RW+puX.

When VMSAv9-128 is not in use, fields Perm[8] to Perm[15] are not used.

This field is not permitted to be cached in a TLB.

When stage 2 Permission Overlay mechanism is disabled, this register is ignored.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing S2POR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, S2POR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nS2POR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = S2POR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = S2POR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = S2POR_EL1;

```

MSR S2POR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nS2POR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            S2POR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.PIEEn == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.PIEEn == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                S2POR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        S2POR_EL1 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

S3_<op1>_<Cn>_<Cm>_<op2>, IMPLEMENTATION DEFINED registers

The S3_<op1>_<Cn>_<Cm>_<op2> characteristics are:

Purpose

This area of the instruction set space is reserved for IMPLEMENTATION DEFINED registers.

Configuration

Each register in this configuration space is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Attributes

- S3_<op1>_<Cn>_<Cm>_<op2> is a:
- 128-bit register when FEAT_SYSREG128 is implemented
 - 64-bit register otherwise

Field descriptions

When FEAT_SYSREG128 is implemented:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	
												IMPLEMENTATION DEFINED																			
												IMPLEMENTATION DEFINED																			
												IMPLEMENTATION DEFINED																			
												IMPLEMENTATION DEFINED																			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
												IMPLEMENTATION DEFINED																			
												IMPLEMENTATION DEFINED																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [127:0]

IMPLEMENTATION DEFINED.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing S3_<op1>_<Cn>_<Cm>_<op2>

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, S3_<op1>_C<Cn>_C<Cm>_<op2>

op0	op1	CRn	CRm	op2
0b11	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysRegRead(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysRegRead(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL2 then
        AArch64.ImpDefSysRegRead(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL3 then
        AArch64.ImpDefSysRegRead(op0, op1, CRn, CRm, op2, t);

```

MSR S3_<op1>_C<Cn>_C<Cm>_<op2>, <Xt>

op0	op1	CRn	CRm	op2
0b11	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysRegWrite(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.ImpDefSysRegWrite(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL2 then
        AArch64.ImpDefSysRegWrite(op0, op1, CRn, CRm, op2, t);
    elsif PSTATE.EL == EL3 then
        AArch64.ImpDefSysRegWrite(op0, op1, CRn, CRm, op2, t);

```

When FEAT_SYSREG128 is implemented

MRRS <Xt+1>, <Xt>, S3_<op1>_C<Cn>_C<Cm>_<op2>

op0	op1	CRn	CRm	op2
0b11	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL2_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x14);
        else
            AArch64.SystemAccessTrap(EL1, 0x14);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL2_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x14);
        elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && (!IsSCTL2_EL1Enabled() ||
SCTL2_EL1.EnIDCP128 == '0') then
            if EL2Enabled() && HCR_EL2.TGE == '1' then
                AArch64.SystemAccessTrap(EL2, 0x14);
            else
                AArch64.SystemAccessTrap(EL1, 0x14);
            elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && (!IsSCTL2_EL2Enabled() ||
SCTL2_EL2.EnIDCP128 == '0') then
                AArch64.SystemAccessTrap(EL2, 0x14);
            elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!IsHCRXEL2Enabled() || HCRX_EL2.EnIDCP128
== '0') then
                AArch64.SystemAccessTrap(EL2, 0x14);
            elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.SystemAccessTrap(EL3, 0x14);
            else
                AArch64.ImpDefSysRegRead128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.EnIDCP128 == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        AArch64.ImpDefSysRegRead128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        AArch64.ImpDefSysRegRead128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL3 then
    AArch64.ImpDefSysRegRead128(op0, op1, CRn, CRm, op2, t, t + 1);

```

When FEAT_SYSREG128 is implemented

MSRR S3_<op1>_C<Cn>_C<Cm>_<op2>, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	op1[2:0]	0b1x11	Cm[3:0]	op2[2:0]

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.TIDCP == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x14);
        else
            AArch64.SystemAccessTrap(EL1, 0x14);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x14);
        elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && (!IsSCTLR2EL1Enabled() ||
SCTLR2_EL1.EnIDCP128 == '0') then
            if EL2Enabled() && HCR_EL2.TGE == '1' then
                AArch64.SystemAccessTrap(EL2, 0x14);
            else
                AArch64.SystemAccessTrap(EL1, 0x14);
            elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && (!IsSCTLR2EL2Enabled() ||
SCTLR2_EL2.EnIDCP128 == '0') then
                AArch64.SystemAccessTrap(EL2, 0x14);
            elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!IsHCRXEL2Enabled() || HCRX_EL2.EnIDCP128
== '0') then
                AArch64.SystemAccessTrap(EL2, 0x14);
            elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.SystemAccessTrap(EL3, 0x14);
            else
                AArch64.ImpDefSysRegWrite128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.EnIDCP128 == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        AArch64.ImpDefSysRegWrite128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnIDCP128 == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.EnIDCP128 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        AArch64.ImpDefSysRegWrite128(op0, op1, CRn, CRm, op2, t, t + 1);
elsif PSTATE.EL == EL3 then
    AArch64.ImpDefSysRegWrite128(op0, op1, CRn, CRm, op2, t, t + 1);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

SCR_EL3, Secure Configuration Register

The SCR_EL3 characteristics are:

Purpose

Defines the configuration of the current Security state. It specifies:

- The Security state of EL0, EL1, and EL2. The Security state is Secure, Non-secure, or Realm.
- The Execution state at lower Exception levels.
- Whether IRQ, FIQ, SError interrupts, and External abort exceptions are taken to EL3.
- Whether various operations are trapped to EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to SCR_EL3 are UNDEFINED.

Attributes

SCR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51				
RES0	NSE	RES0		FGTEn2	GPF	RES0		EnIDCP128	EnTP2	RES0	TRNDR	PFAREn	RES0	TWERRHXEn	RES0	ADEn
TWEDEL	TWEDEn	ECVEn		FGTEn	ATA	EnSCXT	RES0					FIEN	NMEA		EASE	
31	30	29	28	27	26	25	24	23	22	21	20	19				

Bit [63]

Reserved, RES0.

NSE, bit [62] When FEAT_RME is implemented:

This field, evaluated with SCR_EL3.NS, selects the Security state of EL2 and lower Exception levels.

For a description of the values derived by evaluating NS and NSE together, see SCR_EL3.NS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [61:6049]

Reserved, RES0.

FGTEn2, bit [59]**When FEAT_FGT2 is implemented:**

Fine-Grained Traps Enable 2.

When EL2 is implemented, enables the traps to EL2 controlled by [HDFGRTR2_EL2](#), [HDFGWTR2_EL2](#), [HFGITR2_EL2](#), [HFGRTR2_EL2](#), and [HFGWTR2_EL2](#), and controls access to those registers.

FGTEn2	Meaning
0b0	EL2 accesses to the specified registers are trapped to EL3. The values in these registers are treated as 0.
0b1	EL2 accesses to the specified registers are not trapped to EL3 by this mechanism.

Traps caused by accesses to the fine-grained trap registers are reported using an [ESR_ELx](#).EC value of 0x18 and its associated ISS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [58:56]

Reserved, RES0.

EnIDCP128, bit [55]**When FEAT_SYSREG128 is implemented:**

Enables access to IMPLEMENTATION DEFINED 128-bit System registers.

EnIDCP128	Meaning
0b0	Accesses at EL2, EL1, EL0 to IMPLEMENTATION DEFINED 128-bit System registers are trapped to EL3 using an ESR_EL3 .EC value of 0x14, unless the access generates a higher priority exception. Disables the functionality of the 128-bit IMPLEMENTATION DEFINED System registers that are accessible at EL3.
0b1	No accesses are trapped by this control.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [54]

Reserved, RES0.

PFAREn, bit [53]**When FEAT_PFAR is implemented:**

Enable access to Physical Fault Address Registers. When disabled, accesses to Physical Fault Address Registers generate a trap to EL3.

PFAREn	Meaning
0b0	The following instructions at EL2 and EL1 are trapped to EL3: AArch64: MRS and MSR accesses to PFAR_EL1 , PFAR_EL2 , and PFAR_EL12 , reported with EC syndrome value 0x18.
0b1	Accesses of Physical Fault Address Registers are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWERR, bit [52]**When FEAT_RASv2 is implemented:**

Trap writes of error record registers. Enables a trap to EL3 on writes of error record registers.

TWERR	Meaning
0b0	Writes of error record registers are not trapped by this mechanism.
0b1	The following instructions at EL2 and EL1 are trapped to EL3: AArch64: MSR accesses to ERRSELR_EL1 , ERXADDR_EL1 , ERXCTLR_EL1 , ERXMISC0_EL1 , ERXMISC1_EL1 , ERXMISC2_EL1 , ERXMISC3_EL1 , and ERXSTATUS_EL1 , reported with EC syndrome value 0x18. AArch32: MCR accesses to ERRSELR , ERXADDR , ERXADDR2 , ERXCTLR , ERXCTLR2 , ERXMISC0 , ERXMISC1 , ERXMISC2 , ERXMISC3 , ERXMISC4 , ERXMISC5 , ERXMISC6 , ERXMISC7 , and ERXSTATUS , reported with EC syndrome value 0x03.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [51]

Reserved, RES0.

ERR2En, bit [50]**When FEAT_RASv2 is implemented:**

Enable access to additional error record registers. When disabled, accesses to additional error record registers generate a trap to EL3.

Traps MRS accesses to the following instructions at EL2 and EL1 to EL3, reported with the EC syndrome value 0x18:

- [ERXGSR_EL1](#).

ERR2En	Meaning
0b0	The specified instructions at EL2 and EL1 are trapped to EL3:
0b1	Accesses of additional error record registers are not trapped by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MECEn, bit [49]

When FEAT_MEC is implemented:

Enables access to the following EL2 MECID registers, from EL2:

- [MECID_P0_EL2](#).
- [MECID_A0_EL2](#)
- [MECID_P1_EL2](#)
- [MECID_A1_EL2](#)
- [VMECID_P_EL2](#)
- [VMECID_A_EL2](#)

Accesses to these registers are trapped and reported using an ESR_EL3.EC value of 0x18.

MECEn	Meaning
0b0	Accesses from EL2 to a listed MECID register are trapped to EL3. The value of a listed EL2 MECID register is treated as 0 for all purposes other than direct reads or writes to the register from EL3.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

GPF, bit [48]

When FEAT_RME is implemented:

Controls the reporting of Granule protection faults at EL0, EL1 and EL2.

GPF	Meaning
0b0	This control does not cause exceptions to be routed from EL0, EL1 or EL2 to EL3.
0b1	GPFs at EL0, EL1 and EL2 are routed to EL3 and reported as Granule Protection Check exceptions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D128En, Bits bit [47:42]
When FEAT_D128 is implemented:

128-bit System Register trap control. Enables access to 128-bit System Registers via MRRS, MSRR instructions.

- MRRS and MSRR accesses from EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x14:
 - TTBR0_EL1.
 - TTBR1_EL1.
 - RCWMASK_EL1, RCWSMASK_EL1.
 - PAR_EL1.
- MRRS and MSRR accesses from EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x14:
 - TTBR1_EL2 and accesses using the register name TTBR1_EL12.
 - TTBR0_EL2 and accesses using the register name TTBR0_EL12.
 - VTTBR_EL2.

D128En	Meaning
0b0	EL1 and EL2 accesses to the specified registers are disabled, and trapped to EL3.
0b1	This control does not cause any instructions to be trapped.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AIEEn, bit [46]
When FEAT_AIE is implemented:

MAIR2_ELx, AMAIR2_ELx Register access trap control.

- Accesses from EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - AMAIR2_EL1.
 - MAIR2_EL1.
- Accesses from EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - AMAIR2_EL2 and accesses using the register name AMAIR2_EL12.

- [MAIR2_EL2](#) and accesses using the register name MAIR2_EL12.

AIEn	Meaning
0b0	EL1 and EL2 accesses to the specified registers are disabled, and trapped to EL3. The values in these registers are treated as 0.
0b1	This control does not cause any instructions to be trapped.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PIEn, bit [45]

When FEAT_S1PIE is implemented, or FEAT_S2PIE is implemented, or FEAT_S1POE is implemented or FEAT_S2POE is implemented:

Permission Indirection, Overlay Register access trap control. Enables access to Permission Indirection and Overlay registers.

- Accesses from EL0, EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - [POR_EL0](#).
- Accesses from EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - [PIRE0_EL1](#).
 - [PIR_EL1](#).
 - [POR_EL1](#).
 - [S2POR_EL1](#).
- Accesses from EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - [PIRE0_EL2](#) and accesses using the register name PIRE0_EL12.
 - [PIR_EL2](#) and accesses using the register name PIR_EL12.
 - [POR_EL2](#) and accesses using the register name POR_EL12.
 - [S2PIR_EL2](#).

PIEn	Meaning
0b0	EL0, EL1 and EL2 accesses to the specified registers are disabled, and trapped to EL3. The values in these registers are treated as 0.
0b1	This control does not cause any instructions to be trapped.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SCTLR2En, bit [44]**When FEAT_SCTLR2 is implemented:**SCTLR2_ELx register trap control. Enables access to [SCTLR2_EL1](#) and [SCTLR2_EL2](#) registers.

SCTLR2En	Meaning
0b0	EL1 and EL2 accesses to SCTLR2_EL1 and SCTLR2_EL2 registers are disabled, and trapped to EL3. The values in these registers are treated as 0.
0b1	This control does not cause any instructions to be trapped.

Traps are reported using an [ESR_EL3](#).EC value of 0x18.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCR2En, bit [43]**When FEAT_TCR2 is implemented:**TCR2_ELx register trap control. Enables access to [TCR2_EL1](#) and [TCR2_EL2](#) registers.

TCR2En	Meaning
0b0	EL1 and EL2 accesses to TCR2_EL1 and TCR2_EL2 registers are disabled, and trapped to EL3. The values in these registers are treated as 0.
0b1	This control does not cause any instructions to be trapped.

Traps are reported using an [ESR_EL3](#).EC value of 0x18.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RCWMASKEn, bit [42]**When FEAT_THE is implemented:**RCW and RCWS Mask register trap control. Enables access to [RCWMASK_EL1](#), [RCWSMASK_EL1](#).

RCWMASKE _n	Meaning
0b0	EL1 and EL2 accesses to RCWMASK_EL1 and RCWSMASK_EL1 registers are disabled, and trapped to EL3.
0b1	This control does not cause any instructions to be trapped.

Traps for MRS, MSR access are reported using an [ESR_EL3.EC](#) value of 0x18.

Traps for MRRS, MSRR access are reported using an [ESR_EL3.EC](#) value of 0x14.

Traps are not taken if there is a higher priority exception generated by the access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnTP2, bit [41]

When FEAT_SME is implemented:

Traps instructions executed at EL2, EL1, and EL0 that access [TPIDR2_EL0](#) to EL3. The exception is reported using [ESR_ELx.EC](#) value 0x18.

EnTP2	Meaning
0b0	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b1	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TRNDR, bit [40]

When FEAT_RNG_TRAP is implemented:

Controls trapping of reads of [RNDR](#) and [RNDRRS](#). The exception is reported using [ESR_ELx.EC](#) value 0x18.

TRNDR	Meaning
0b0	This control does not cause RNDR and RNDRRS to be trapped. When FEAT_RNG is implemented: <ul style="list-style-type: none"> ID_AA64ISAR0_EL1.RNDR returns the value 0b0001. When FEAT_RNG is not implemented: <ul style="list-style-type: none"> ID_AA64ISAR0_EL1.RNDR returns the value 0b0000. MRS reads of RNDR and RNDRRS are UNDEFINED.
0b1	ID_AA64ISAR0_EL1 .RNDR returns the value 0b0001. Any attempt to read RNDR or RNDRRS is trapped to EL3.

When FEAT_RNG is not implemented, Arm recommends that SCR_EL3.TRNDR is initialized before entering Exception levels below EL3 and not subsequently changed.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [39]

Reserved, RES0.

HXEn, bit [38]**When FEAT_HCX is implemented:**

Enables access to the [HCRX_EL2](#) register at EL2 from EL3.

HXEn	Meaning
0b0	Accesses at EL2 to HCRX_EL2 are trapped to EL3. Indirect reads of HCRX_EL2 return 0.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ADEn, bit [37]**When FEAT_LS64_ACCDATA is implemented:**

Enables access to the [ACCDATA_EL1](#) register at EL1 and EL2.

ADEn	Meaning
0b0	Accesses to ACCDATA_EL1 at EL1 and EL2 are trapped to EL3, unless the accesses are trapped to EL2 by the EL2 fine-grained trap.
0b1	This control does not cause accesses to ACCDATA_EL1 to be trapped.

If the [HFGWTR_EL2](#).nACCDATA_EL1 or [HFGTR_EL2](#).nACCDATA_EL1 traps are enabled, they take priority over this trap.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnAS0, bit [36]**When FEAT_LS64_ACCDATA is implemented:**

Traps execution of an ST64BV0 instruction at EL0, EL1, or EL2 to EL3.

EnAS0	Meaning
0b0	EL0 execution of an ST64BV0 instruction is trapped to EL3, unless it is trapped to EL1 by SCTLR_EL1.EnAS0 , or to EL2 by either HCRX_EL2.EnAS0 or SCTLR_EL2.EnAS0 . EL1 execution of an ST64BV0 instruction is trapped to EL3, unless it is trapped to EL2 by HCRX_EL2.EnAS0 . EL2 execution of an ST64BV0 instruction is trapped to EL3.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV0 instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000001.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AMVOFFEN, bit [35]

When FEAT_AMUv1p1 is implemented:

Activity Monitors Virtual Offsets Enable.

AMVOFFEN	Meaning
0b0	Accesses to AMEVCNTVOFF0<n>_EL2 and AMEVCNTVOFF1<n>_EL2 at EL2 are trapped to EL3. Indirect reads of the virtual offset registers are zero.
0b1	Accesses to AMEVCNTVOFF0<n>_EL2 and AMEVCNTVOFF1<n>_EL2 are not affected by this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TME, bit [34]

When FEAT_TME is implemented:

Enables access to the TSTART, TCOMMIT, TTEST and TCANCEL instructions at EL0, EL1 and EL2.

TME	Meaning
0b0	EL0, EL1 and EL2 accesses to TSTART, TCOMMIT, TTEST and TCANCEL instructions are UNDEFINED.
0b1	This control does not cause any instruction to be UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWEDEL, bits [33:30]**When FEAT_TWED is implemented:**

TWE Delay. A 4-bit unsigned number that, when SCR_EL3.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE* caused by SCR_EL3.TWE as $2^{(\text{TWEDEL} + 8)}$ cycles.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWEDEn, bit [29]**When FEAT_TWED is implemented:**

TWE Delay Enable. Enables a configurable delayed trap of the WFE* instruction caused by SCR_EL3.TWE.

Traps are reported using an ESR_ELx.EC value of 0x01.

TWEDEn	Meaning
0b0	The delay for taking the trap is IMPLEMENTATION DEFINED.
0b1	The delay for taking the trap is at least the number of cycles defined in SCR_EL3.TWEDEL.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ECVEn, bit [28]**When FEAT_ECV is implemented:**

ECV Enable. Enables access to the [CNTPOFF_EL2](#) register.

ECVEn	Meaning
0b0	EL2 accesses to CNTPOFF_EL2 are trapped to EL3, and the value of CNTPOFF_EL2 is treated as 0 for all purposes other than direct reads or writes to the register from EL3.
0b1	EL2 accesses to CNTPOFF_EL2 are not trapped to EL3 by this mechanism.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

FGTEn, bit [27]**When FEAT_FGT is implemented:**

Fine-Grained Traps Enable. When EL2 is implemented, enables the traps to EL2 controlled by [HAFGRTR_EL2](#), [HDFGRTR_EL2](#), [HDFGWTR_EL2](#), [HFGTRTR_EL2](#), [HFGITR_EL2](#), and [HFGWTR_EL2](#), and controls access to those registers.

Note

If EL2 is not implemented but EL3 is implemented, FEAT_FGT implements the [MDCR_EL3](#).TDC traps.

FGTEn	Meaning
0b0	EL2 accesses to HAFGRTR_EL2 , HDFGRTR_EL2 , HDFGWTR_EL2 , HFGTRTR_EL2 , HFGITR_EL2 and HFGWTR_EL2 registers are trapped to EL3, and the traps to EL2 controlled by those registers are disabled.
0b1	EL2 accesses to HAFGRTR_EL2 , HDFGRTR_EL2 , HDFGWTR_EL2 , HFGTRTR_EL2 , HFGITR_EL2 and HFGWTR_EL2 registers are not trapped to EL3 by this mechanism.

Traps caused by accesses to the fine-grained trap registers are reported using an ESR_ELx.EC value of 0x18 and its associated ISS.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ATA, bit [26]**When FEAT_MTE2 is implemented:**

Allocation Tag Access. Controls access to Allocation Tags, System registers for Memory tagging, and prevention of Tag checking, at EL2, EL1 and EL0.

ATA	Meaning
0b0	Access to Allocation Tags is prevented at EL2, EL1, and EL0. Accesses at EL1 and EL2 to GCR_EL1 , RGSr_EL1 , TFSR_EL1 , TFSR_EL2 or TFSRE0_EL1 that are not UNDEFINED or trapped to a lower Exception level are trapped to EL3. Accesses at EL2 using MRS or MSR with the register name TFSR_EL12 that are not UNDEFINED are trapped to EL3. Memory accesses at EL2, EL1, and EL0 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL2, EL1, and EL0. This control does not prevent Tag checking at EL2, EL1, and EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnSCXT, bit [25]**When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:**

Enables ~~Enable~~ access to the [SCXTNUM_EL2](#), [SCXTNUM_EL1](#), and [SCXTNUM_EL0](#) registers.

EnSCXT	Meaning
0b0	Accesses at EL0, EL1 and EL2 to SCXTNUM_EL0 , SCXTNUM_EL1 , or SCXTNUM_EL2 registers are trapped to EL3 if they are not trapped by a higher priority exception, and the values of these registers are treated as 0.
0b1	This control does not cause any accesses to be trapped, or register values to be treated as 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [24:22]

Reserved, RES0.

FIEN, bit [21]**When FEAT_RASv1p1 is implemented:**

Fault Injection enable. Trap accesses to the registers [ERXPFPCDN_EL1](#), [ERXPFPCCTL_EL1](#), and [ERXPFPCGF_EL1](#) from EL1 and EL2 to EL3, reported using an ESR_ELx.EC value of 0x18.

FIEN	Meaning
0b0	Accesses to the specified registers from EL1 and EL2 generate a Trap exception to EL3.
0b1	This control does not cause any instructions to be trapped.

If EL3 is not implemented, the Effective value of SCR_EL3.FIEN is 0b1.

If [ERRIDR_EL1](#).NUM is zero, meaning no error records are implemented, or no error record accessible using System registers is owned by a node that implements the RAS Common Fault Injection Model Extension, then this bit might be RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NMEA, bit [20]**When FEAT_DoubleFault is implemented:**

Non-maskable External Aborts. Controls whether PSTATE.A masks SError exceptions at EL3. When [SCR_EL3.EA](#) == 1, controls whether PSTATE.A masks SError interrupts at EL3.

NMEA	Meaning
0b0	Error exceptions are not taken at EL3 if PSTATE.A == 1. If SCR_EL3.EA == 1, asserted SError interrupts are not taken at EL3 if PSTATE.A == 1.
0b1	Error exceptions are taken at EL3 regardless of the value of PSTATE.A. If SCR_EL3.EA == 1, asserted SError interrupts are taken at EL3 regardless of the value of PSTATE.A.

This When field SCR_EL3.EA is == ignored by the PE and treated as zero when all of the following are true 0:

- This field is ignored and its Effective value is 0.
- SCR_EL3.EA Asserted == SError 0. interrupts are not taken at EL3 regardless of the value of PSTATE.A and this field.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EASE, bit [19]

When FEAT_DoubleFault is implemented:

External aborts to SError interrupt vector.

EASE	Meaning
0b0	Synchronous External abort exceptions taken to EL3 are taken to the appropriate synchronous exception vector offset from VBAR_EL3.
0b1	Synchronous External abort exceptions taken to EL3 are taken to the appropriate SError interrupt vector offset from VBAR_EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EEL2, bit [18]

When FEAT_SEL2 is implemented:

Secure EL2 Enable.

EEL2	Meaning
0b0	All behaviors associated with Secure EL2 are disabled. All registers, including timer registers, defined by FEAT_SEL2 are UNDEFINED, and those timers are disabled.
0b1	All behaviors associated with Secure EL2 are enabled.

When the value of this bit is 1, then:

- When SCR_EL3.NS == 0, the SCR_EL3.RW bit is treated as 1 for all purposes other than reading or writing the register.
- If Secure EL1 is using AArch32, then any of the following operations, executed in Secure EL1, is trapped to Secure EL2, using the EC value of ESR_EL2.EC == 0x3 :
 - A read or write of the SCR.

- A read or write of the [NSACR](#).
 - A read or write of the [MVBAR](#).
 - A read or write of the [SDCR](#).
 - Execution of an `ATS12NSO**` instruction.
- If Secure EL1 is using AArch32, then any of the following operations, executed in Secure EL1, is trapped to Secure EL2 using the EC value of [ESR_EL2](#).EC == 0x0 :
 - Execution of an SRS instruction that uses `R13_mon`.
 - Execution of an MRS (Banked register) or MSR (Banked register) instruction that would access [SPSR_mon](#), `R13_mon`, or `R14_mon`.

Note

If the Effective value of `SCR_EL3.EEL2` is 0, then these operations executed in Secure EL1 using AArch32 are trapped to EL3.

A Secure only implementation that does not implement EL3 but implements EL2, behaves as if `SCR_EL3.EEL2` == 1.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

API, bit [17]
When FEAT_SEL2 is implemented and FEAT_PAuth is implemented:

Controls the use of the following instructions related to Pointer Authentication. Traps are reported using an `ESR_ELx.EC` value of 0x09:

- PACGA, which is always enabled.
- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZB, RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ, ERETAA, ERETAB, LDRAA and LDRAZ when:
 - In EL0, when [HCR_EL2.TGE](#) == 0 or [HCR_EL2.E2H](#) == 0, and the associated [SCTLR_EL1.En<N><M>](#) == 1.
 - In EL0, when [HCR_EL2.TGE](#) == 1 and [HCR_EL2.E2H](#) == 1, and the associated [SCTLR_EL2.En<N><M>](#) == 1.
 - In EL1, when the associated [SCTLR_EL1.En<N><M>](#) == 1.
 - In EL2, when the associated [SCTLR_EL2.En<N><M>](#) == 1.

API	Meaning
0b0	The use of any instruction related to pointer authentication in any Exception level except EL3 when the instructions are enabled are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2 .API bit.
0b1	This control does not cause any instructions to be trapped.

An instruction is trapped only if Pointer Authentication is enabled for that instruction, for more information, see '[PACSystem generation register and control verification of keys pointer authentication](#)'.

Note

If FEAT_PAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_SEL2 is not implemented and FEAT_PAuth is implemented:

Controls the use of instructions related to Pointer Authentication:

- PACGA.
- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZ, RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ, ERETAA, ERETAB, LDRAA and LDRAB when:
 - In Non-secure EL0, when [HCR_EL2.TGE](#) == 0 or [HCR_EL2.E2H](#) == 0, and the associated [SCTLR_EL1.En<N><M>](#) == 1.
 - In Non-secure EL0, when [HCR_EL2.TGE](#) == 1 and [HCR_EL2.E2H](#) == 1, and the associated [SCTLR_EL2.En<N><M>](#) == 1.
 - In Secure EL0, when the associated [SCTLR_EL1.En<N><M>](#) == 1.
 - In Secure or Non-secure EL1, when the associated [SCTLR_EL1.En<N><M>](#) == 1.
 - In EL2, when the associated [SCTLR_EL2.En<N><M>](#) == 1.

API	Meaning
0b0	The use of any instruction related to pointer authentication in any Exception level except EL3 when the instructions are enabled are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.API bit.
0b1	This control does not cause any instructions to be trapped.

Note

If FEAT_PAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

APK, bit [16]

When FEAT_PAuth is implemented:

Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers, using an ESR_ELx.EC value of 0x18, from EL1 or EL2 to EL3 unless they are trapped to EL2 as a result of the [HCR_EL2.APK](#) bit or other traps:

- [APIAKeyLo_EL1](#), [APIAKeyHi_EL1](#), [APIBKeyLo_EL1](#), [APIBKeyHi_EL1](#).
- [APDAKeyLo_EL1](#), [APDAKeyHi_EL1](#), [APDBKeyLo_EL1](#), [APDBKeyHi_EL1](#).
- [APGAKeyLo_EL1](#), and [APGAKeyHi_EL1](#).

APK	Meaning
0b0	Access to the registers holding "key" values for pointer authentication from EL1 or EL2 are trapped to EL3 unless they are trapped to EL2 as a result of the HCR_EL2.APK bit or other traps.
0b1	This control does not cause any instructions to be trapped.

For more information, see '[PACSystem generation register and control verification of keys pointer authentication](#)'.

Note

If FEAT_PAAuth is implemented but EL3 is not implemented, the system behaves as if this bit is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TERR, bit [15]

When FEAT_RAS is implemented:

Trap Error record accesses. Accesses to the RAS ERR* and RAS ERX* registers from EL1 and EL2 to EL3 are trapped as follows:

- Accesses from EL1 and EL2 using AArch64 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x18:
 - [ERRIDR_EL1](#), [ERRSELR_EL1](#), [ERXADDR_EL1](#), [ERXCTLR_EL1](#), [ERXFR_EL1](#), [ERXMISC0_EL1](#), [ERXMISC1_EL1](#), and [ERXSTATUS_EL1](#).
- If FEAT_RASv1p1 is implemented, accesses from EL1 and EL2 using AArch64 to [ERXMISC2_EL1](#), and [ERXMISC3_EL1](#), are trapped and reported using an ESR_ELx.EC value of 0x18.
- Accesses from EL1 and EL2 using AArch32, to the following registers are trapped and reported using an ESR_ELx.EC value of 0x03:
 - [ERRIDR](#), [ERRSELR](#), [ERXADDR](#), [ERXADDR2](#), [ERXCTLR](#), [ERXCTLR2](#), [ERXFR](#), [ERXFR2](#), [ERXMISC0](#), [ERXMISC1](#), [ERXMISC2](#), [ERXMISC3](#), and [ERXSTATUS](#).
- If FEAT_RASv1p1 is implemented, accesses from EL1 and EL2 using AArch32 to the following registers are trapped and reported using an ESR_ELx.EC value of 0x03:
 - [ERXMISC4](#), [ERXMISC5](#), [ERXMISC6](#), and [ERXMISC7](#).

TERR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Accesses to the specified registers from EL1 and EL2 generate a Trap exception to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TLOR, bit [14]

When FEAT_LOR is implemented:

Trap LOR registers. Traps accesses to the [LORSA_EL1](#), [LOREA_EL1](#), [LORN_EL1](#), [LORC_EL1](#), and [LORID_EL1](#) registers from EL1 and EL2 to EL3, unless the access has been trapped to EL2.

TLOR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 and EL2 accesses to the LOR registers that are not UNDEFINED are trapped to EL3, unless it is trapped HCR_EL2.TLOR .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWE, bit [13]

Traps EL2, EL1, and EL0 execution of WFE instructions to EL3, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFET instruction.

TWE	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFE instruction at any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWE , HCR.TWE , SCTLR_EL1.nTWE , SCTLR_EL2.nTWE , or HCR_EL2.TWE .

In AArch32 state, the attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

For more information about when WFE instructions can cause the PE to enter a low-power state, see 'Wait for Event mechanism and Send event'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TWI, bit [12]

Traps EL2, EL1, and EL0 execution of WFI instructions to EL3, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFIT instruction.

TWI	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFI instruction at any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWI , HCR.TWI , SCTLR_EL1.nTWI , SCTLR_EL2.nTWI , or HCR_EL2.TWI .

In AArch32 state, the attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the

WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

For more information about when WFI instructions can cause the PE to enter a low-power state, see 'Wait for Interrupt'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ST, bit [11]

Traps Secure EL1 accesses to the Counter-timer Physical Secure timer registers to EL3, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

ST	Meaning
0b0	Secure EL1 using AArch64 accesses to the CNTPS_TVAL_EL1 , CNTPS_CTL_EL1 , and CNTPS_CVAL_EL1 are trapped to EL3 when Secure EL2 is disabled. If Secure EL2 is enabled, the behavior is as if the value of this field was 0b1.
0b1	This control does not cause any instructions to be trapped.

Note

Accesses to the Counter-timer Physical Secure timer registers are always enabled at EL3. These registers are not accessible at EL0.

When FEAT_RME is implemented and Secure state is not implemented, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

RW, bit [10]

When EL1 is capable of using AArch32 or EL2 is capable of using AArch32:

Execution state control for lower Exception levels.

RW	Meaning
0b0	Lower levels are all AArch32.
0b1	The next lower level is AArch64. If EL2 is present: <ul style="list-style-type: none"> • EL2 is AArch64. • EL2 controls EL1 and EL0 behaviors. If EL2 is not present: <ul style="list-style-type: none"> • EL1 is AArch64. • EL0 is determined by the Execution state described in the current process state when executing at EL0.

If AArch32 state is supported by the implementation at EL1, SCR_EL3.NS == 1 and AArch32 state is not supported by the implementation at EL2, the Effective value of this bit is 1.

If AArch32 state is supported by the implementation at EL1, FEAT_SEL2 is implemented and SCR_EL3.{EEL2, NS} == {1, 0}, the Effective value of this bit is 1.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RAO/WI.

SIF, bit [9]

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction execution from memory marked in the first stage of translation as being Non-secure.

SIF	Meaning
0b0	Secure state instruction execution from memory marked in the first stage of translation as being Non-secure is permitted.
0b1	Secure state instruction execution from memory marked in the first stage of translation as being Non-secure is not permitted.

When FEAT_RME is implemented and Secure state is not implemented, this bit is RES0.

When FEAT_PAN3 is implemented, it is IMPLEMENTATION DEFINED whether SCR_EL3.SIF is also used to determine instruction access permission for the purpose of PAN.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

HCE, bit [8]

Hypervisor Call instruction enable. Enables HVC instructions at EL3 and, if EL2 is enabled in the current Security state, at EL2 and EL1, in both Execution states, reported using an ESR_ELx.EC value of 0x00.

HCE	Meaning
0b0	HVC instructions are UNDEFINED.
0b1	HVC instructions are enabled at EL3, EL2, and EL1.

Note

HVC instructions are always UNDEFINED at EL0 and, if Secure EL2 is disabled, at Secure EL1. Any resulting exception is taken from the current Exception level to the current Exception level.

If EL2 is not implemented, this bit is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SMD, bit [7]

Secure Monitor Call disable. Disables SMC instructions at EL1 and above, from any Security state and both Execution states, reported using an ESR_ELx.EC value of 0x00.

SMD	Meaning
0b0	SMC instructions are enabled at EL3, EL2 and EL1.
0b1	SMC instructions are UNDEFINED.

Note

SMC instructions are always UNDEFINED at EL0. Any resulting exception is taken from the current Exception level to the current Exception level.

If [HCR_EL2.TSC](#) or [HCR.TSC](#) traps attempted EL1 execution of SMC instructions to EL2, that trap has priority over this disable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

Bits [5:4]

Reserved, RES1.

EA, bit [3]

External Abort and SError interrupt routing.

EA	Meaning
0b0	When executing at Exception levels below EL3, External aborts and SError interrupts are not taken to EL3. In addition, when executing at EL3: <ul style="list-style-type: none"> • SError interrupts are not taken. • External aborts are taken to EL3.
0b1	When executing at any Exception level, External aborts and SError interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

FIQ, bit [2]

Physical FIQ Routing.

FIQ	Meaning
0b0	When executing at Exception levels below EL3, physical FIQ interrupts are not taken to EL3. When executing at EL3, physical FIQ interrupts are not taken.
0b1	When executing at any Exception level, physical FIQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRQ, bit [1]

Physical IRQ Routing.

IRQ	Meaning
0b0	When executing at Exception levels below EL3, physical IRQ interrupts are not taken to EL3. When executing at EL3, physical IRQ interrupts are not taken.
0b1	When executing at any Exception level, physical IRQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NS, bit [0]**When FEAT_RME is implemented:**

Non-secure bit. This field is used in combination with SCR_EL3.NSE to select the Security state of EL2 and lower Exception levels.

NSE	NS	Meaning
0b0	0b0	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Reserved.
0b1	0b1	Realm.

When Secure state is not implemented, SCR_EL3.NS is RES1 and its effective value is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Non-secure bit.

NS	Meaning
0b0	Indicates that EL0 and EL1 are in Secure state. When FEAT_SEL2 is implemented and SCR_EL3.EEL2 == 1, then EL2 is using AArch64 and in Secure state.
0b1	Indicates that Exception levels lower than EL3 are in Non-secure state, so memory accesses from those Exception levels cannot access Secure memory.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SCR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCR_EL3;

```

MSR SCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SCR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

no old file

htmldiff from-

(new)

SCTLR2_EL1, System Control Register (EL1)

The SCTLR2_EL1 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL1 and EL0.

Configuration

This register is present only when FEAT_SCTLR2 is implemented. Otherwise, direct accesses to SCTLR2_EL1 are UNDEFINED.

Attributes

SCTLR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39		38		37		36		35		34	33	32	
RES0																																					
RES0																										EnIDCP128				RES0		EnANERR		EnADERR		RES0	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7		6		5		4		3		2	1	0	

Bits [63:7]

Reserved, RES0.

EnIDCP128, bit [6]

When FEAT_SYSREG128 is implemented:

Enables access to IMPLEMENTATION DEFINED 128-bit System registers.

EnIDCP128	Meaning
0b0	Accesses at EL0 to IMPLEMENTATION DEFINED 128-bit System registers are trapped to EL1 using an ESR_EL1.EC value of 0x14, unless the access generates a higher priority exception. Disables the functionality of the 128-bit IMPLEMENTATION DEFINED System registers that are accessible at EL1.
0b1	No accesses are trapped by this control.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [5]

Reserved, RES0.

EnANERR, bit [4]**When FEAT_ANERR is implemented:**

Enable Asynchronous Normal Read Error.

EnANERR	Meaning
0b0	External abort on Normal memory reads generate synchronous Data Abort exceptions in the EL1&0 translation regime.
0b1	External abort on Normal memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL1&0 translation regime.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Normal memory read in every case.

Setting this field to 0 might have a performance impact for Normal memory reads.

This field is ignored by the PE and treated as zero when any of the following are true:

- All of the following are true:
 - EL2 is implemented and enabled in the current Security state.
 - The Effective value of [HCRX_EL2](#).SCTLR2En is 0.
- All of the following are true:
 - EL2 is implemented and enabled in the current Security state.
 - The Effective value of [HCRX_EL2](#).EnSNERR is 1.
- [SCR_EL3](#).SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnADERR, bit [3]**When FEAT_ADERR is implemented:**

Enable Asynchronous Device Read Error.

EnADERR	Meaning
0b0	External abort on Device memory reads generate synchronous Data Abort exceptions in the EL1&0 translation regime.
0b1	External abort on Device memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL1&0 translation regime.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.

- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Device memory read in every case.

Setting this field to 0 might have a performance impact for Device memory reads.

This field is ignored by the PE and treated as zero when any of the following are true:

- All of the following are true:
 - EL2 is implemented and enabled in the current Security state.
 - The Effective value of [HCRX_EL2](#).SCTLR2En is 0.
- All of the following are true:
 - EL2 is implemented and enabled in the current Security state.
 - The Effective value of [HCRX_EL2](#).EnSDERR is 1.
- [SCR_EL3](#).SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [2:0]

Reserved, RES0.

Accessing SCTLR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x278];
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR2_EL2;
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL1;

```

MSR SCTLR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x278] = X[t, 64];
        else
            SCTLR2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            SCTLR2_EL2 = X[t, 64];
        else
            SCTLR2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        SCTLR2_EL1 = X[t, 64];

```

MRS <Xt>, SCTLR2_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x278];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = SCTLR2_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR2_EL1;
    else
        UNDEFINED;

```

MSR SCTLR2_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x278] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SCTLR2_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SCTLR2_EL1 = X[t, 64];
    else
        UNDEFINED;

```

no old file

htmldiff from-

(new)

SCTLR2_EL2, System Control Register (EL2)

The SCTLR2_EL2 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL2.

When FEAT_VHE is implemented and the value of [HCR_EL2](#).{E2H,TGE} is {1,1}, these controls also apply to execution at EL0.

Configuration

This register is present only when FEAT_SCTLR2 is implemented. Otherwise, direct accesses to SCTLR2_EL2 are UNDEFINED.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

SCTLR2_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															

Bits [63:7]

Reserved, RES0.

EnIDCP128, bit [6]

When FEAT_SYSREG128 is implemented and HCR_EL2.[E2H,TGE] != 0b11:

Enables access to IMPLEMENTATION DEFINED 128-bit System registers.

EnIDCP128	Meaning
0b0	Accesses at EL0 to IMPLEMENTATION DEFINED 128-bit System registers are trapped to EL2 using an ESR_EL2.EC value of 0x14, unless the access generates a higher priority exception. Disables the functionality of the 128-bit IMPLEMENTATION DEFINED System registers that are accessible at EL2.
0b1	No accesses are trapped by this control.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [5]

Reserved, RES0.

EnANERR, bit [4]**When FEAT_ANERR is implemented:**

Enable Asynchronous Normal Read Error.

EnANERR	Meaning
0b0	External abort on Normal memory reads generate synchronous Data Abort exceptions in the EL2 and EL2&0 translation regimes.
0b1	External abort on Normal memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL2 and EL2&0 translation regimes.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Normal memory read in every case.

Setting this field to 0 might have a performance impact for Normal memory reads.

This field is ignored by the PE and treated as zero when [SCR_EL3.SCTLR2En](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnADERR, bit [3]**When FEAT_ADERR is implemented:**

Enable Asynchronous Device Read Error.

EnADERR	Meaning
0b0	External abort on Device memory reads generate synchronous Data Abort exceptions in the EL2 and EL2&0 translation regimes.
0b1	External abort on Device memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL2 and EL2&0 translation regimes.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.

- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Device memory read in every case.

Setting this field to 0 might have a performance impact for Device memory reads.

This field is ignored by the PE and treated as zero when [SCR_EL3](#).SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [2]

Reserved, RES0.

MECRL, bit [1]

When FEAT_MEC is implemented:

Enables MEC for the Realm physical address space at EL2.

MECRL	Meaning
0b0	MEC is not enabled for the Realm physical address space at EL2.
0b1	MEC is enabled for the Realm physical address space at EL2.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [0]

Reserved, RES0.

Accessing SCTLR2_EL2

When FEAT_VHE is implemented, and [HCR_EL2](#).E2H is 1, without explicit synchronization, accesses from EL2 using the register name SCTLR2_EL2 or SCTLR2_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SCTLR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL2;

```

MSR SCTLR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCTLR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR2_EL2 = X[t, 64];

```

MRS <Xt>, SCTLR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x278];
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR2_EL2;
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL1;

```

MSR SCTLR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x278] = X[t, 64];
    else
        SCTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        SCTLR2_EL2 = X[t, 64];
    else
        SCTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR2_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SCTLR2_EL3, System Control Register (EL3)

The SCTLR2_EL3 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL3.

Configuration

This register is present only when FEAT_SCTLR2 is implemented. Otherwise, direct accesses to SCTLR2_EL3 are UNDEFINED.

Attributes

SCTLR2_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															

Bits [63:5]

Reserved, RES0.

EnANERR, bit [4]

When FEAT_ANERR is implemented:

Enable Asynchronous Normal Read Error.

EnANERR	Meaning
0b0	External abort on Normal memory reads generate synchronous Data Abort exceptions in the EL3 translation regime.
0b1	External abort on Normal memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL3 translation regime.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Normal memory read in every case.

Setting this field to 0 might have a performance impact for Normal memory reads.

This field is ignored by the PE and treated as zero when [SCR_EL3](#).SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnADERR, bit [3]

When FEAT_ADERR is implemented:

Enable Asynchronous Device Read Error.

EnADERR	Meaning
0b0	External abort on Device memory reads generate synchronous Data Abort exceptions in the EL3 translation regime.
0b1	External abort on Device memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL3 translation regime.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Device memory read in every case.

Setting this field to 0 might have a performance impact for Device memory reads.

This field is ignored by the PE and treated as zero when [SCR_EL3.SCTLR2En](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [2]

Reserved, RES0.

MECRL, bit [1]

When FEAT_MEC is implemented:

Enables MEC for the Realm physical address space at EL3.

MECRL	Meaning
0b0	MEC is not enabled for the Realm physical address space at EL3.
0b1	MEC is enabled for the Realm physical address space at EL3.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [0]

Reserved, RES0.

Accessing SCTLR2_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR2_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL3;
```

MSR SCTLR2_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SCTLR2_EL3 = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SCTLR_EL1, System Control Register (EL1)

The SCTLR_EL1 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL1 and EL0.

Configuration

AArch64 System register SCTLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [SCTLR\[31:0\]](#).

Attributes

SCTLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53
TIDCP	SPINTMASK	NMI	EnTP2	TCSORES0	TCSO0EPAN	EPAN	EnALS	EnALS	EnAS0	EnAS0
EnIA	EnIB	LSMAOE	nTLSMD	EnDA	UCI	EE	E0E	SPAN	EIS	IESB
31	30	29	28	27	26	25	24	23	22	21

TIDCP, bit [63]
When FEAT_TIDCP1 is implemented:

Trap IMPLEMENTATION DEFINED functionality. When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps EL0 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL1.

TIDCP	Meaning
0b0	No instructions accessing the System register or System instruction spaces are trapped by this mechanism.
0b1	Instructions accessing the following System register or System instruction spaces are trapped to EL1 by this mechanism: <ul style="list-style-type: none">In AArch64 state, EL0 access to the encodings in the following reserved encoding spaces are trapped and reported using EC syndrome 0x18:<ul style="list-style-type: none">IMPLEMENTATION DEFINED System instructions, which are accessed using SYS and SYSL, with CRn == {11, 15}.IMPLEMENTATION DEFINED System registers, which are accessed using MRS and MSR with the S3_<op1>_<Cn>_<Cm>_<op2> register name.In AArch32 state, EL0 MCR and MRC access to the following encodings are trapped and reported using EC syndrome 0x03:<ul style="list-style-type: none">All coproc==p15, CRn==c9, opc1 == {0-7}, CRm == {c0-c2, c5-c8}, opc2 == {0-7}.All coproc==p15, CRn==c10, opc1 =={0-7}, CRm == {c0, c1, c4, c8}, opc2 == {0-7}.All coproc==p15, CRn==c11, opc1=={0-7}, CRm == {c0-c8, c15}, opc2 == {0-7}.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SPINTMASK, bit [62]

When FEAT_NMI is implemented:

SP Interrupt Mask enable. When SCTLR_EL1.NMI is 1, controls whether PSTATE.SP acts as an interrupt mask, and controls the value of PSTATE.ALLINT on taking an exception to EL1.

SPINTMASK	Meaning
0b0	Does not cause PSTATE.SP to mask interrupts. PSTATE.ALLINT is set to 1 on taking an exception to EL1.
0b1	When PSTATE.SP is 1 and execution is at EL1, an IRQ or FIQ interrupt that is targeted to EL1 is masked regardless of any denotation of Superpriority. PSTATE.ALLINT is set to 0 on taking an exception to EL1.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NMI, bit [61]

When FEAT_NMI is implemented:

Non-maskable Interrupt enable.

NMI	Meaning
0b0	This control does not affect interrupt masking behavior.
0b1	This control enables all of the following: <ul style="list-style-type: none"> The use of the PSTATE.ALLINT interrupt mask. IRQ and FIQ interrupts to have Superpriority as an additional attribute. PSTATE.SP to be used as an interrupt mask.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to 0.

Otherwise:

Reserved, RES0.

EnTP2, bit [60]**When FEAT_SME is implemented:**

Traps instructions executed at EL0 that access [TPIDR2_EL0](#) to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1. The exception is reported using ESR_ELx.EC value 0x18.

EnTP2	Meaning
0b0	This control causes execution of these instructions at EL0 to be trapped.
0b1	This control does not cause execution of any instructions to be trapped.

If FEAT_VHE is implemented, EL2 is implemented and enabled in the current Security state, and [HCR_EL2.{E2H, TGE}](#) == {1, 1}, this field has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCSO, Bits bit [59:58]**When FEAT_MTE_STORE_ONLY is implemented:**

Tag Checking Store Only.

TCSO	Meaning
0b0	This field has no effect on Tag checking.
0b1	Load instructions executed in EL1 are Tag Unchecked.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCSO0, bit [58]**When FEAT_MTE_STORE_ONLY is implemented:**

When [HCR_EL2.{E2H, TGE}](#) != {1, 1}, Tag Checking Store Only in EL0.

TCSO0	Meaning
0b0	This field has no effect on Tag checking.
0b1	Load instructions executed in EL0 are Tag Unchecked.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EPAN, bit [57]**When FEAT_PAN3 is implemented:**

Enhanced Privileged Access Never. When PSTATE.PAN is 1, determines whether an EL1 data access to a page with stage 1 EL0 instruction access permission generates a Permission fault as a result of the Privileged Access Never mechanism.

EPAN	Meaning
0b0	No additional Permission faults are generated by this mechanism.
0b1	An EL1 data access to a page with stage 1 EL0 data access permission or stage 1 EL0 instruction access permission generates a Permission fault. Any speculative data accesses that would generate a Permission fault if the accesses were not speculative will not cause an allocation into a cache.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnALS, bit [56]**When FEAT_LS64 is implemented:**

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an LD64B or ST64B instruction at EL0 to EL1.

EnALS	Meaning
0b0	Execution of an LD64B or ST64B instruction at EL0 is trapped to EL1.
0b1	This control does not cause any instructions to be trapped.

A trap of an LD64B or ST64B instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000002.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnAS0, bit [55]**When FEAT_LS64_ACCDATA is implemented:**

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an ST64BV0 instruction at EL0 to EL1.

EnAS0	Meaning
0b0	Execution of an ST64BV0 instruction at EL0 is trapped to EL1.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV0 instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000001.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnASR, bit [54]

When FEAT_LS64_V is implemented:

When [HCR_EL2](#).{E2H, TGE} != {1, 1}, traps execution of an ST64BV instruction at EL0 to EL1.

EnASR	Meaning
0b0	Execution of an ST64BV instruction at EL0 is trapped to EL1.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000000.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TME, bit [53]

When FEAT_TME is implemented:

Enables the Transactional Memory Extension at EL1.

TME	Meaning
0b0	Any attempt to execute a TSTART instruction at EL1 is trapped to EL1, unless HCR_EL2 .TME or SCR_EL3 .TME causes TSTART instructions to be UNDEFINED at EL1.
0b1	This control does not cause any TSTART instruction to be trapped.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TME0, bit [52]

When FEAT_TME is implemented:

Enables the Transactional Memory Extension at EL0.

TME0	Meaning
0b0	Any attempt to execute a TSTART instruction at EL0 is trapped to EL1, unless HCR_EL2.TME or SCR_EL3.TME causes TSTART instructions to be UNDEFINED at EL0.
0b1	This control does not cause any TSTART instruction to be trapped.

If FEAT_VHE is implemented, EL2 is implemented and enabled in the current Security state, and [HCR_EL2.{E2H, TGE}](#) == {1, 1}, this field has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TMT, bit [51]

When FEAT_TME is implemented:

Forces a trivial implementation of the Transactional Memory Extension at EL1.

TMT	Meaning
0b0	This control does not cause any TSTART instruction to fail.
0b1	When the TSTART instruction is executed at EL1, the transaction fails with a TRIVIAL failure cause.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TMT0, bit [50]

When FEAT_TME is implemented:

Forces a trivial implementation of the Transactional Memory Extension at EL0.

TMT0	Meaning
0b0	This control does not cause any TSTART instruction to fail.
0b1	When the TSTART instruction is executed at EL0, the transaction fails with a TRIVIAL failure cause.

If FEAT_VHE is implemented, EL2 is implemented and enabled in the current Security state, and [HCR_EL2.{E2H, TGE}](#) == {1, 1}, this field has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWEDEL, bits [49:46]**When FEAT_TWED is implemented:**

TWE Delay. A 4-bit unsigned number that, when SCTLR_EL1.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE* caused by SCTLR_EL1.nTWE as $2^{(TWEDEL + 8)}$ cycles.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TWEDEn, bit [45]**When FEAT_TWED is implemented:**

TWE Delay Enable. Enables a configurable delayed trap of the WFE* instruction caused by SCTLR_EL1.nTWE.

TWEDEn	Meaning
0b0	The delay for taking the trap is IMPLEMENTATION DEFINED.
0b1	The delay for taking the trap is at least the number of cycles defined in SCTLR_EL1.TWEDEL.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DSSBS, bit [44]**When FEAT_SSBS is implemented:**

Default PSTATE.SSBS value on Exception Entry.

DSSBS	Meaning
0b0	PSTATE.SSBS is set to 0 on an exception to EL1.
0b1	PSTATE.SSBS is set to 1 on an exception to EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Otherwise:

Reserved, RES0.

ATA, bit [43]**When FEAT_MTE2 is implemented:**

Allocation Tag Access in EL1.

When [SCR_EL3.ATA](#) == 1 and [HCR_EL2.ATA](#) == 1, controls access to Allocation Tags and Tag Check operations in EL1.

ATA	Meaning
0b0	Access to Allocation Tags is prevented at EL1. Memory accesses at EL1 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL1. Tag Checked memory accesses at EL1 are subject to a Tag Check operation. The Tag Check operation depends on the type of tag at the memory being accessed: <ul style="list-style-type: none"> For Allocation Tagged memory, an Allocation Tag Check operation. If FEAT_MTE_CANONICAL_TAGS is implemented, for Canonically Tagged memory, a Canonical Tag Check operation.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL1, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ATA0, bit [42]

When FEAT_MTE2 is implemented:

Allocation Tag Access in EL0.

When [SCR_EL3.ATA](#) == 1, [HCR_EL2.ATA](#) == 1, and [HCR_EL2.{E2H, TGE}](#) != {1, 1}, controls access to Allocation Tags and Tag Check operations in EL0.

ATA0	Meaning
0b0	Access to Allocation Tags is prevented at EL0. Memory accesses at EL0 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL0. Tag Checked memory accesses at EL0 are subject to a Tag Check operation. The Tag Check operation depends on the type of tag at the memory being accessed: <ul style="list-style-type: none"> For Allocation Tagged memory, an Allocation Tag Check operation. If FEAT_MTE_CANONICAL_TAGS is implemented, for Canonically Tagged memory, a Canonical Tag Check operation.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL1, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCF, bits [41:40]**When FEAT_MTE2 is implemented:**

Tag Check Fault in EL1. Controls the effect of Tag Check Faults due to Loads and Stores in EL1.

If FEAT_MTE3 is not implemented, the value 0b11 is reserved.

TCF	Meaning	Applies when
0b00	Tag Check Faults have no effect on the PE.	When FEAT_MTE3 is implemented
0b01	Tag Check Faults cause a synchronous exception.	
0b10	Tag Check Faults are asynchronously accumulated.	
0b11	Tag Check Faults cause a synchronous exception on reads, and are asynchronously accumulated on writes.	

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCF0, bits [39:38]**When FEAT_MTE2 is implemented:**

Tag Check Fault in EL0. When [HCR_EL2](#).{E2H,TGE} != {1,1}, controls the effect of Tag Check Faults due to Loads and Stores in EL0.

If FEAT_MTE3 is not implemented, the value 0b11 is reserved.

Note

Software may change this control bit on a context switch.

TCF0	Meaning	Applies when
0b00	Tag Check Faults have no effect on the PE.	When FEAT_MTE3 is implemented
0b01	Tag Check Faults cause a synchronous exception.	
0b10	Tag Check Faults are asynchronously accumulated.	
0b11	Tag Check Faults cause a synchronous exception on reads, and are asynchronously accumulated on writes.	

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ITFSB, bit [37]**When FEAT_MTE2 is implemented:**

When synchronous exceptions are not being generated by Tag Check Faults, this field controls whether on exception entry into EL1, all Tag Check Faults due to instructions executed before exception entry, that are reported asynchronously, are synchronized into [TFSRE0_EL1](#) and [TFSR_EL1](#) registers.

ITFSB	Meaning
0b0	Tag Check Faults are not synchronized on entry to EL1.
0b1	Tag Check Faults are synchronized on entry to EL1.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT1, bit [36]**When FEAT_BT1 is implemented:**

PAC Branch Type compatibility at EL1.

BT1	Meaning
0b0	When the PE is executing at EL1, PACIASP and PACIBSP are compatible with PSTATE.BTYPE == 0b11.
0b1	When the PE is executing at EL1, PACIASP and PACIBSP are not compatible with PSTATE.BTYPE == 0b11.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT0, bit [35]**When FEAT_BT1 is implemented:**

PAC Branch Type compatibility at EL0.

BT0	Meaning
0b0	When the PE is executing at EL0, PACIASP and PACIBSP are compatible with PSTATE.BTYPE == 0b11.
0b1	When the PE is executing at EL0, PACIASP and PACIBSP are not compatible with PSTATE.BTYPE == 0b11.

When the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, the value of SCTLR_EL1.BT0 has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [34]

Reserved, RES0.

MSCEn, bit [33]**When FEAT_MOPS is implemented and (HCR_EL2.E2H == 0 or HCR_EL2.TGE == 0):**

Memory Copy and Memory Set instructions Enable. Enables execution of the Memory Copy and Memory Set instructions at EL0.

MSCEn	Meaning
0b0	Execution of the Memory Copy and Memory Set instructions is UNDEFINED at EL0.
0b1	This control does not cause any instructions to be UNDEFINED.

When FEAT_MOPS is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 1}, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CMOW, bit [32]**When FEAT_CMOW is implemented:**

Controls cache maintenance instruction permission for the following instructions executed at EL0.

- [IC IVAU](#), [DC CIVAC](#), [DC CIGDVAC](#) and [DC CIGVAC](#).

CMOW	Meaning
0b0	These instructions executed at EL0 with stage 1 read permission, but without stage 1 write permission, do not generate a stage 1 permission fault.
0b1	If enabled as a result of SCTLR_EL1 .UCI==1, these instructions executed at EL0 with stage 1 read permission, but without stage 1 write permission, generate a stage 1 permission fault.

When AArch64.HCR_EL2.{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

For this control, stage 1 has write permission if all of the following apply:

- AP[2] is 0 or DBM is 1 in the stage 1 descriptor.
- Where APTTable is in use, APTTable[1] is 0 for all levels of the translation table.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnIA, bit [31]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnIA	Meaning
0b0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is enabled.

Note

This field controls the behavior of the AddPACIA and AuthIA pseudocode functions. Specifically, when the field is 1, AddPACIA returns a copy of a pointer to which a pointer authentication code has been added, and AuthIA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnIB, bit [30]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnIB	Meaning
0b0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is enabled.

Note

This field controls the behavior of the AddPACIB and AuthIB pseudocode functions. Specifically, when the field is 1, AddPACIB returns a copy of a pointer to which a pointer authentication code has been added, and AuthIB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LSMAOE, bit [29]**When FEAT_LSMAOC is implemented:**

Load Multiple and Store Multiple Atomicity and Ordering Enable.

LSMAOE	Meaning
0b0	For all memory accesses at EL0, A32 and T32 Load Multiple and Store Multiple can have an interrupt taken during the sequence memory accesses, and the memory accesses are not required to be ordered.
0b1	The ordering and interrupt behavior of A32 and T32 Load Multiple and Store Multiple at EL0 is as defined for Armv8.0.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

nTLSMD, bit [28]**When FEAT_LSMAOC is implemented:**

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory.

nTLSMD	Meaning
0b0	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are trapped and generate a stage 1 Alignment fault.
0b1	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are not trapped.

This bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

EnDA, bit [27]**When FEAT_PAAuth is implemented:**

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnDA	Meaning
0b0	Pointer authentication (using the APDAKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDAKey_EL1 key) of data addresses is enabled.

Note

This field controls the behavior of the AddPACDA and AuthDA pseudocode functions. Specifically, when the field is 1, AddPACDA returns a copy of a pointer to which a pointer authentication code has been added, and AuthDA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UCI, bit [26]

Traps EL0 execution of cache maintenance instructions, to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

This applies to [DC CVAU](#), [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#), and [IC IVAU](#).

If FEAT_DPB2 is implemented, this trap also applies to [DC CVADP](#).

If FEAT_MTE is implemented, this trap also applies to [DC CIGVAC](#), [DC CIGDVAC](#), [DC CGVAC](#), [DC CGDVAC](#), [DC CGVAP](#), and [DC CGDVAP](#).

If FEAT_DPB2 and FEAT_MTE are implemented, this trap also applies to [DC CGVADP](#) and [DC CGDVADP](#).

UCI	Meaning
0b0	Execution of the specified instructions at EL0 using AArch64 is trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on execution at EL0.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

EE, bit [25]

Endianness of data accesses at EL1, and stage 1 translation table walks in the EL1&0 translation regime.

EE	Meaning
0b0	Explicit data accesses at EL1, and stage 1 translation table walks in the EL1&0 translation regime are little-endian.
0b1	Explicit data accesses at EL1, and stage 1 translation table walks in the EL1&0 translation regime are big-endian.

If an implementation does not provide Big-endian support at Exception levels higher than EL0, this bit is RES0.

If an implementation does not provide Little-endian support at Exception levels higher than EL0, this bit is RES1.

The EE bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

EOE, bit [24]

Endianness of data accesses at EL0.

EOE	Meaning
0b0	Explicit data accesses at EL0 are little-endian.
0b1	Explicit data accesses at EL0 are big-endian.

If an implementation only supports Little-endian accesses at EL0, then this bit is RES0. This option is not permitted when SCTLR_EL1.EE is RES1.

If an implementation only supports Big-endian accesses at EL0, then this bit is RES1. This option is not permitted when SCTLR_EL1.EE is RES0.

This bit has no effect on the endianness of LDTR, LDTRH, LDTRSH, LDTRSW, STTR, and STTRH instructions executed at EL1.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

SPAN, bit [23]

When FEAT_PAN is implemented:

Set Privileged Access Never, on taking an exception to EL1.

SPAN	Meaning
0b0	PSTATE.PAN is set to 1 on taking an exception to EL1.
0b1	The value of PSTATE.PAN is left unchanged on taking an exception to EL1.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

EIS, bit [22]**When FEAT_ExS is implemented:**

Exception Entry is Context Synchronizing.

EIS	Meaning
0b0	The taking of an exception to EL1 is not a context synchronizing event.
0b1	The taking of an exception to EL1 is a context synchronizing event.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

If SCTLR_EL1.EIS is set to 0b0:

- Indirect writes to [ESR_EL1](#), [FAR_EL1](#), [SPSR_EL1](#), [ELR_EL1](#) are synchronized on exception entry to EL1, so that a direct read of the register after exception entry sees the indirectly written value caused by the exception entry.
- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL1.EIS:

- Changes to the PSTATE information on entry to EL1.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores and data processing instructions.
- Exit from Debug state.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1~~, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

IESB, bit [21]**When FEAT_IESB is implemented:**

Implicit Error Synchronization event enable. Possible values are:

IESB	Meaning
0b0	Disabled.
0b1	An implicit error synchronization event is added: <ul style="list-style-type: none"> • At each exception taken to EL1. • Before the operational pseudocode of each ERET instruction executed at EL1.

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and its Effective value might be 0 or 1 regardless of the value of the field. If the Effective value of the field is 1, then an implicit error synchronization event is added after each DCPSX instruction taken to EL1 and before each DRPS instruction executed at EL1, in addition to the other cases where it is added.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TSCXT, bit [20]

When FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented:

Trap EL0 Access to the [SCXTNUM_EL0](#) register, when EL0 is using AArch64.

TSCXT	Meaning
0b0	EL0 access to SCXTNUM_EL0 is not disabled by this mechanism.
0b1	EL0 access to SCXTNUM_EL0 is disabled, causing an exception to EL1, or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2 .TGE is 1. The value of SCXTNUM_EL0 is treated as 0.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

WXN, bit [19]

Write permission implies XN (Execute-never). For the EL1&0 translation regime, this bit can force all memory regions that are writable to be treated as XN.

WXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable in the EL1&0 translation regime is forced to XN for accesses from software executing at EL1 or EL0.

This bit applies only when SCTLR_EL1.M bit is set.

The WXN bit is permitted to be cached in a TLB.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

nTWE, bit [18]

Traps EL0 execution of WFE instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, from both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFET instruction.

nTWE	Meaning
0b0	Any attempt to execute a WFE instruction at EL0 is trapped, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

In AArch32 state, the attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

Traps EL0 execution of WFI instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, from both Execution states, reported using an ESR_ELx.EC value of 0x01.

When FEAT_WFxT is implemented, this trap also applies to the WFIT instruction.

nTWI	Meaning
0b0	Any attempt to execute a WFI instruction at EL0 is trapped, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

In AArch32 state, the attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

UCT, bit [15]

Traps EL0 accesses to the [CTR_EL0](#) to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

UCT	Meaning
0b0	Accesses to the CTR_EL0 from EL0 using AArch64 are trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

DZE, bit [14]

Traps EL0 execution of [DC ZVA](#) instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

If FEAT_MTE is implemented, this trap also applies to [DC GVA](#) and [DC GZVA](#).

DZE	Meaning
0b0	Any attempt to execute an instruction that this trap applies to at EL0 using AArch64 is trapped. Reading DCZID_EL0 .DZP from EL0 returns 1, indicating that the instructions this trap applies to are not supported.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

EnDB, bit [13]

When FEAT_PAuth is implemented:

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnDB	Meaning
0b0	Pointer authentication (using the APDBKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDBKey_EL1 key) of data addresses is enabled.

Note

This field controls the behavior of the AddPACDB and AuthDB pseudocode functions. Specifically, when the field is 1, AddPACDB returns a copy of a pointer to which a pointer authentication code has been added, and AuthDB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

I, bit [12]

Stage 1 instruction access Cacheability control, for accesses at EL0 and EL1:

I	Meaning
0b0	All instruction access to Stage 1 Normal memory from EL0 and EL1 are Stage 1 Non-cacheable. If the value of SCTLR_EL1.M is 0, instruction accesses from stage 1 of the EL1&0 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
0b1	This control has no effect on the Stage 1 Cacheability of instruction access to Stage 1 Normal memory from EL0 and EL1. If the value of SCTLR_EL1.M is 0, instruction accesses from stage 1 of the EL1&0 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

When the value of the [HCR_EL2.DC](#) bit is 1, then instruction access to Normal memory from EL0 and EL1 are Cacheable regardless of the value of the SCTLR_EL1.I bit.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, in a system where the PE resets into EL1, this field resets to 0.

EOS, bit [11]**When FEAT_ExS is implemented:**

Exception Exit is Context Synchronizing.

EOS	Meaning
0b0	An exception return from EL1 is not a context synchronizing event
0b1	An exception return from EL1 is a context synchronizing event

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1,1}, this bit has no effect on execution at EL0.

If SCTLR_EL1.EOS is set to 0b0:

- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL1.EOS:

- The indirect write of the PSTATE and PC values from [SPSR_EL1](#) and [ELR_EL1](#) on exception return is synchronized.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores and data processing instructions.
- Exit from Debug state.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL1, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

EnRCTX, bit [10]**When FEAT_SPECRES is implemented:**

Enable EL0 ~~access~~Access to the following **System** instructions:

- **CFPRCTX, DVPRCTX and CPPRCTX** ~~AArch32 CFPRCTX, DVPRCTX and CPPRCTX~~ instructions.
- ~~If AArch64 FEAT_SPECRES2 CFP is RCTX, implemented, DVP RCT and CPP RCTX instructions. COSPRCTX.~~
- **CFP RCTX, DVP RCTX and CPP RCTX** instructions.
- **If FEAT_SPECRES2 is implemented, COSP RCTX.**

EnRCTX	Meaning
0b0	EL0 access to these instructions is disabled, and these instructions are trapped to EL1, or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2.TGE is 1.
0b1	EL0 access to these instructions is enabled.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1,1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1,~~ this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UMA, bit [9]

User Mask Access. Traps EL0 execution of MSR and MRS instructions that access the PSTATE.{D, A, I, F} masks to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.

UMA	Meaning
0b0	Any attempt at EL0 using AArch64 to execute an MRS, MSR(REGISTER), or MSR(IMMEDIATE) instruction that accesses the DAIF is trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, ~~in a system where the PE resets into EL1,~~ this field resets to an architecturally UNKNOWN value.

SED, bit [8]**When EL0 is capable of using AArch32:**

SETEND instruction disable. Disables SETEND instructions at EL0 using AArch32.

SED	Meaning
0b0	SETEND instruction execution is enabled at EL0 using AArch32.
0b1	SETEND instructions are UNDEFINED at EL0 using AArch32 and any attempt at EL0 to access a SETEND instruction generates an exception to EL1, or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2.TGE is 1, reported using an ESR_ELx.EC value of 0x00.

If the implementation does not support mixed-endian operation at any Exception level, this bit is RES1.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

ITD, bit [7]

When EL0 is capable of using AArch32:

IT Disable. Disables some uses of IT instructions at EL0 using AArch32.

ITD	Meaning
0b0	All IT instruction functionality is enabled at EL0 using AArch32.
0b1	Any attempt at EL0 using AArch32 to execute any of the following is UNDEFINED and generates an exception, reported using an ESR_ELx.EC value of 0x00, to EL1 or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2.TGE is 1: <ul style="list-style-type: none"> All encodings of the IT instruction with hw1[3:0] != 1000. All encodings of the subsequent instruction with the following values for hw1: <ul style="list-style-type: none"> 0b11xxxxxxxxxxxx: All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM. 0b1011xxxxxxxxxxxx: All instructions in 'Miscellaneous 16-bit instructions'. 0b10100xxxxxxxxxxx: ADD Rd, PC, #imm 0b01001xxxxxxxxxxx: LDR Rd, [PC, #imm] 0b0100x1xxx1111xxx: ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC. 0b010001xx1xxxx111: ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers unpredictable cases with BLX Rn. <p>These instructions are always UNDEFINED, regardless of whether they would pass or fail the condition code check that applies to them as a result of being in an IT block.</p> <p>It is IMPLEMENTATION DEFINED whether the IT instruction is treated as:</p> <ul style="list-style-type: none"> A 16-bit instruction, that can only be followed by another 16-bit instruction. The first half of a 32-bit instruction. <p>This means that, for the situations that are UNDEFINED, either the second 16-bit instruction or the 32-bit instruction is UNDEFINED.</p> <p>An implementation might vary dynamically as to whether IT is treated as a 16-bit instruction or the first half of a 32-bit instruction.</p>

If an instruction in an active IT block that would be disabled by this field sets this field to 1 then behavior is CONSTRAINED UNPREDICTABLE. For more information, see 'Changes to an ITD control by an instruction in an IT block'.

ITD is optional, but if it is implemented in the SCTLR_EL1 then it must also be implemented in the [SCTLR_EL2](#), [HSCTLR](#), and [SCTLR](#).

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

When an implementation does not implement ITD, access to this field is **RAZ/WI**.

Otherwise:

Reserved, RES1.

nAA, bit [6]

When FEAT_LSE2 is implemented:

Non-aligned access. This bit controls generation of Alignment faults at EL1 and EL0 under certain conditions.

The following instructions generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH.
- STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH.

The following instructions generate an Alignment fault if all bytes being accessed for a single register are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDIAPP, LDAPR, STILP, and STLR.
- If Advanced SIMD and floating-point instructions are implemented, LDDAPUR, LDAP1, STLUR, and STL1.

nAA	Meaning
0b0	Unaligned accesses by the specified instructions generate an Alignment fault.
0b1	This control does not generate Alignment faults.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, **in a system where the PE resets into EL1**, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CP15BEN, bit [5]

When EL0 is capable of using AArch32:

System instruction memory barrier enable. Enables accesses to the DMB, DSB, and ISB System instructions in the (coproc==0b1111) encoding space from EL0:

CP15BEN	Meaning
0b0	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED and generates an exception to EL1, or to EL2 when it is implemented and enabled for the current Security state and HCR_EL2.TGE is 1. The exception is reported using an ESR_ELx.EC value of 0x00.
0b1	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

CP15BEN is optional, but if it is implemented in the SCTLR_EL1 then it must also be implemented in the [SCTLR_EL2](#), [HSCTLR](#), and [SCTLR](#).

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

When an implementation does not implement CP15BEN, access to this field is **RAO/WI**.

Otherwise:

Reserved, RES0.

SA0, bit [4]

SP Alignment check enable for EL0. When set to 1, if a load or store instruction executed at EL0 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see 'SP alignment checking'.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

SA, bit [3]

SP Alignment check enable. When set to 1, if a load or store instruction executed at EL1 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see 'SP alignment checking'.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- On a Warm reset, [in a system where the PE resets into EL1](#), this field resets to an architecturally UNKNOWN value.

C, bit [2]

Stage 1 Cacheability control, for data accesses.

C	Meaning
0b0	All data access to Stage 1 Normal memory from EL0 and EL1, and all Normal memory accesses from unified cache to the EL1&0 Stage 1 translation tables, are treated as Stage 1 Non-cacheable.
0b1	This control has no effect on the Stage 1 Cacheability of: <ul style="list-style-type: none"> Data access to Normal memory from EL0 and EL1. Normal memory accesses to the EL1&0 Stage 1 translation tables.

When the **Effective** value of the [HCR_EL2.DC](#) bit in the current Security state is 1, the PE ignores [SCTLR_EL1.C.SCTLR.C](#). This means that **Non-secure** EL0 and **Non-secure** EL1 data accesses to Normal memory are Cacheable.

When FEAT_VHE is implemented, and the **Effective** value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, in a system where the PE resets into EL1, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL1 and EL0.

A	Meaning
0b0	Alignment fault checking disabled when executing at EL1 or EL0. Instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, do not check that the address being accessed is aligned to the size of the data element(s) being accessed.
0b1	Alignment fault checking enabled when executing at EL1 or EL0. All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have an alignment check regardless of the value of the A bit.

If FEAT_MOPS is implemented, SETG* instructions have an alignment check regardless of the value of the A bit.

When FEAT_VHE is implemented, and the value of [HCR_EL2.{E2H, TGE}](#) is {1, 1}, this bit has no effect on execution at EL0.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL1, this field resets to an architecturally UNKNOWN value.

M, bit [0]

MMU enable for EL1&0 stage 1 address translation.

M	Meaning
0b0	EL1&0 stage 1 address translation disabled. See the SCTLR_EL1.I field for the behavior of instruction accesses to Normal memory.
0b1	EL1&0 stage 1 address translation enabled.

If the **Effective** value of **HCR_EL2**.{DC, TGE} **in the current Security state** is not {0, 0} then **in Non-secure state** the PE behaves as if the value of the SCTLR_EL1.M field is 0 for all purposes other than returning the value of a direct read of the field.

When FEAT_VHE is implemented, and the **Effective** value of **HCR_EL2**.{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

The reset behavior of this field is:

- **On a Warm reset:**
 - **When EL2 is not implemented and EL3 is not implemented, this field resets to 0.**
 - **Otherwise, this field resets to an architecturally UNKNOWN value.**
- **On a Warm reset, in a system where the PE resets into EL1, this field resets to 0.**

Accessing SCTLR_EL1

When **HCR_EL2**.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic SCTLR_EL1 or SCTLR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x110];
    else
        X[t, 64] = SCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR_EL2;
    else
        X[t, 64] = SCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR_EL1;

```

MSR SCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x110] = X[t, 64];
    else
        SCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SCTLR_EL2 = X[t, 64];
    else
        SCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR_EL1 = X[t, 64];

```

MRS <Xt>, SCTLR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x110];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR_EL1;
    else
        UNDEFINED;

```

MSR SCTLR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x110] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SCTLR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SCTLR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

SCTLR_EL2, System Control Register (EL2)

The SCTLR_EL2 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL2.

When FEAT_VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, these controls apply also to execution at EL0.

Configuration

AArch64 System register SCTLR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HSCTLR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

SCTLR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53
TIDCP	SPINTMASK	NMI	EnTP2	TCSORES0	TCSO0EPAN	EPANEnALS	EnALSENAS0	EnAS0EnASR	EnASRTMET	TME
EnIA	EnIB	LSMAOE	nTLSMD	EnDA	UCI	EE	E0E	SPAN	EIS	IESB
31	30	29	28	27	26	25	24	23	22	21

TIDCP, bit [63]
When FEAT_TIDCP1 is implemented and HCR_EL2.E2H == 1:

Trap IMPLEMENTATION DEFINED functionality. Traps EL0 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL2.

TIDCP	Meaning
0b0	No instructions accessing the System register or System instruction spaces are trapped by this mechanism.
0b1	<p>If HCR_EL2.TGE==0, no instructions accessing the System register or System instruction spaces are trapped by this mechanism.</p> <p>If HCR_EL2.TGE==1, instructions accessing the following System register or System instruction spaces are trapped to EL2 by this mechanism:</p> <ul style="list-style-type: none"> In AArch64 state, EL0 access to the encodings in the following reserved encoding spaces are trapped and reported using EC syndrome 0x18: <ul style="list-style-type: none"> IMPLEMENTATION DEFINED System instructions, which are accessed using SYS and SYSL, with CRn == {11, 15}. IMPLEMENTATION DEFINED System registers, which are accessed using MRS and MSR with the S3_<op1>_<Cn>_<Cm>_<op2> register name. In AArch32 state, EL0 MCR and MRC access to the following encodings are trapped and reported using EC syndrome 0x03: <ul style="list-style-type: none"> All coproc==p15, CRn==c9, opc1 == {0-7}, CRm == {c0-c2, c5-c8}, opc2 == {0-7}. All coproc==p15, CRn==c10, opc1 == {0-7}, CRm == {c0, c1, c4, c8}, opc2 == {0-7}. All coproc==p15, CRn==c11, opc1 == {0-7}, CRm == {c0-c8, c15}, opc2 == {0-7}.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SPINTMASK, bit [62]

When FEAT_NMI is implemented:

SP Interrupt Mask enable. When SCTLR_EL2.NMI is 1, controls whether PSTATE.SP acts as an interrupt mask, and controls the value of PSTATE.ALLINT on taking an exception to EL2.

SPINTMASK	Meaning
0b0	Does not cause PSTATE.SP to mask interrupts. PSTATE.ALLINT is set to 1 on taking an exception to EL2.
0b1	When PSTATE.SP is 1 and execution is at EL2, an IRQ or FIQ interrupt that is targeted to EL2 is masked regardless of any denotation of Superpriority. PSTATE.ALLINT is set to 0 on taking an exception to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NMI, bit [61]**When FEAT_NMI is implemented:**

Non-maskable Interrupt enable.

NMI	Meaning
0b0	This control does not affect interrupt masking behavior.
0b1	This control enables all of the following: <ul style="list-style-type: none"> The use of the PSTATE.ALLINT interrupt mask. IRQ and FIQ interrupts to have Superpriority as an additional attribute. PSTATE.SP to be used as an interrupt mask.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Otherwise:

Reserved, RES0.

EnTP2, bit [60]**When FEAT_SME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Traps instructions executed at EL0 that access [TPIDR2_EL0](#) to EL2 when EL2 is implemented and enabled for the current Security state. The exception is reported using ESR_ELx.EC value 0x18.

EnTP2	Meaning
0b0	This control causes execution of these instructions at EL0 to be trapped.
0b1	This control does not cause execution of any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_SME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TCSO, Bits bit [59:58]**When FEAT_MTE_STORE_ONLY is implemented:**

Tag Checking Store Only.

TCSO	Meaning
0b0	This field has no effect on Tag checking.
0b1	Load instructions executed in EL2 are Tag Unchecked.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCSO0, bit [58]

When FEAT_MTE_STORE_ONLY is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Tag Checking Store Only in EL0.

TCSO0	Meaning
0b0	This field has no effect on Tag checking.
0b1	Load instructions executed in EL0 are Tag Unchecked.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EPAN, bit [57]

When FEAT_PAN3 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Enhanced Privileged Access Never. When PSTATE.PAN is 1, determines whether an EL2 data access to a page with EL0 instruction access permission generates a Permission fault as a result of the Privileged Access Never mechanism.

EPAN	Meaning
0b0	No additional Permission faults are generated by this mechanism.
0b1	An EL2 data access to a page with stage 1 EL0 data access permission or stage 1 EL0 instruction access permission generates a Permission fault. Any speculative data accesses that would generate a Permission fault if the accesses were not speculative will not cause an allocation into a cache.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_PAN3 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

EnALS, bit [56]

When FEAT_LS64 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Traps execution of an LD64B or ST64B instruction at EL0 to EL2.

EnALS	Meaning
0b0	Execution of an LD64B or ST64B instruction at EL0 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an LD64B or ST64B instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000002.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

EnAS0, bit [55]

When FEAT_LS64_ACCDATA is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Traps execution of an ST64BV0 instruction at EL0 to EL2.

EnAS0	Meaning
0b0	Execution of an ST64BV0 instruction at EL0 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV0 instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000001.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64_ACCDATA is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

EnASR, bit [54]

When FEAT_LS64_V is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Traps execution of an ST64BV instruction at EL0 to EL2.

EnASR	Meaning
0b0	Execution of an ST64BV instruction at EL0 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

A trap of an ST64BV instruction is reported using an ESR_ELx.EC value of 0x0A, with an ISS code of 0x0000000.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_LS64_V is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TME, bit [53]

When FEAT_TME is implemented:

Enables the Transactional Memory Extension at EL2.

TME	Meaning
0b0	Any attempt to execute a TSTART instruction at EL2 is trapped, unless HCR_EL2.TME or SCR_EL3.TME causes TSTART instructions to be UNDEFINED at EL2.
0b1	This control does not cause any TSTART instruction to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TME0, bit [52]

When FEAT_TME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Enables the Transactional Memory Extension at EL0.

TME0	Meaning
0b0	Any attempt to execute a TSTART instruction at EL0 is trapped to EL2, unless HCR_EL2.TME or SCR_EL3.TME causes TSTART instructions to be UNDEFINED at EL0.
0b1	This control does not cause any TSTART instruction to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_TME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TMT, bit [51]**When FEAT_TME is implemented:**

Forces a trivial implementation of the Transactional Memory Extension at EL2.

TMT	Meaning
0b0	This control does not cause any TSTART instruction to fail.
0b1	When the TSTART instruction is executed at EL2, the transaction fails with a TRIVIAL failure cause.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TMT0, bit [50]**When FEAT_TME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Forces a trivial implementation of the Transactional Memory Extension at EL0.

TMT0	Meaning
0b0	This control does not cause any TSTART instruction to fail.
0b1	When the TSTART instruction is executed at EL0, the transaction fails with a TRIVIAL failure cause.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_TME is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TWEDEL, bits [49:46]**When FEAT_TWED is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

TWE Delay. A 4-bit unsigned number that, when SCTLR_EL2.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE caused by SCTLR_EL2.nTWE as $2^{(TWEDEL + 8)}$ cycles.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_TWED is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TWEDn, bit [45]

When FEAT_TWED is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

TWE Delay Enable. Enables a configurable delayed trap of the WFE instruction caused by SCTLR_EL2.nTWE.

TWEDn	Meaning
0b0	The delay for taking a WFE trap is IMPLEMENTATION DEFINED.
0b1	The delay for taking a WFE trap is at least the number of cycles defined in SCTLR_EL2.TWEDEL.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_TWED is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

DSSBS, bit [44]

When FEAT_SSBS is implemented:

Default PSTATE.SSBS value on Exception Entry.

DSSBS	Meaning
0b0	PSTATE.SSBS is set to 0 on an exception to EL2.
0b1	PSTATE.SSBS is set to 1 on an exception to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Otherwise:

Reserved, RES0.

ATA, bit [43]

When FEAT_MTE2 is implemented:

Allocation Tag Access in EL2.

When [SCR_EL3.ATA](#) is 1, controls access to Allocation Tags and Tag Check operations in EL2.

ATA	Meaning
0b0	Access to Allocation Tags is prevented at EL2. Memory accesses at EL2 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL2. Tag Checked memory accesses at EL2 are subject to a Tag Check operation. The Tag Check operation depends on the type of tag at the memory being accessed: <ul style="list-style-type: none"> For Allocation Tagged memory, an Allocation Tag Check operation. If FEAT_MTE_CANONICAL_TAGS is implemented, for Canonically Tagged memory, a Canonical Tag Check operation.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

ATA0, bit [42]

When FEAT_MTE2 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Allocation Tag Access in EL0.

When [SCR_EL3.ATA](#) is 1, controls access to Allocation Tags and Tag Check operations in EL0.

ATA0	Meaning
0b0	Access to Allocation Tags is prevented at EL0. Memory accesses at EL0 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL0. Tag Checked memory accesses at EL0 are subject to a Tag Check operation. The Tag Check operation depends on the type of tag at the memory being accessed: <ul style="list-style-type: none"> For Allocation Tagged memory, an Allocation Tag Check operation. If FEAT_MTE_CANONICAL_TAGS is implemented, for Canonically Tagged memory, a Canonical Tag Check operation.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_MTE2 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

TCF, bits [41:40]**When FEAT_MTE2 is implemented:**

Tag Check Fault in EL2. Controls the effect of Tag Check Faults due to Loads and Stores in EL2.

TCF	Meaning	Applies when
0b00	Tag Check Faults have no effect on the PE.	
0b01	Tag Check Faults cause a synchronous exception.	
0b10	Tag Check Faults are asynchronously accumulated.	
0b11	Tag Check Faults cause a synchronous exception on reads, and are asynchronously accumulated on writes.	When FEAT_MTE3 is implemented

If FEAT_MTE3 is not implemented, the value 0b11 is reserved.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCF0, bits [39:38]**When FEAT_MTE2 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Tag Check Fault in EL0. Controls the effect of Tag Check Faults due to Loads and Stores in EL0.

TCF0	Meaning	Applies when
0b00	Tag Check Faults have no effect on the PE.	
0b01	Tag Check Faults cause a synchronous exception.	
0b10	Tag Check Faults are asynchronously accumulated.	
0b11	Tag Check Faults cause a synchronous exception on reads, and are asynchronously accumulated on writes.	When FEAT_MTE3 is implemented

If FEAT_MTE3 is not implemented, the value 0b11 is reserved.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_MTE2 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

ITFSB, bit [37]**When FEAT_MTE2 is implemented:**

When synchronous exceptions are not being generated by Tag Check Faults, this field controls whether on exception entry into EL2, all Tag Check Faults due to instructions executed before exception entry, that are reported asynchronously, are synchronized into [TFSRE0_EL1](#), [TFSR_EL1](#) and [TFSR_EL2](#) registers.

ITFSB	Meaning
0b0	Tag Check Faults are not synchronized on entry to EL2.
0b1	Tag Check Faults are synchronized on entry to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT, bit [36]**When FEAT_BTI is implemented:**

PAC Branch Type compatibility at EL2.

When [HCR_EL2](#).{E2H, TGE} == {1, 1}, this bit is named BT1.

BT	Meaning
0b0	When the PE is executing at EL2, PACIASP and PACIBSP are compatible with PSTATE.BTYPE == 0b11.
0b1	When the PE is executing at EL2, PACIASP and PACIBSP are not compatible with PSTATE.BTYPE == 0b11.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT0, bit [35]**When FEAT_BTI is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

PAC Branch Type compatibility at EL0.

BT0	Meaning
0b0	When the PE is executing at EL0, PACIASP and PACIBSP are compatible with PSTATE.BTYPE == 0b11.
0b1	When the PE is executing at EL0, PACIASP and PACIBSP are not compatible with PSTATE.BTYPE == 0b11.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_BTI is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

Bit [34]

Reserved, RES0.

MSCEn, bit [33]

When FEAT_MOPS is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Memory Copy and Memory Set instructions Enable. Enables execution of the Memory Copy and Memory Set instructions at EL0.

MSCEn	Meaning
0b0	Execution of the Memory Copy and Memory Set instructions is UNDEFINED at EL0.
0b1	This control does not cause any instructions to be UNDEFINED.

When FEAT_MOPS is implemented and [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the Effective value of this bit is 0b1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_MOPS is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

CMOW, bit [32]

When FEAT_CMOW is implemented and HCR_EL2.E2H == 1:

Controls cache maintenance instruction permission for the following instructions executed at EL0.

- [IC IVAU](#), [DC CIVAC](#), [DC CIGDVAC](#) and [DC CIGVAC](#).

CMOW	Meaning
0b0	These instructions executed at EL0 with stage 1 read permission, but without stage 1 write permission, do not generate a stage 1 permission fault.
0b1	If enabled as a result of SCTLR_EL2.UCI =1, these instructions executed at EL0 with stage 1 read permission, but without stage 1 write permission, generate a stage 1 permission fault.

When [HCR_EL2.TGE](#) is 0, this bit has no effect on execution at EL0.

For this control, stage 1 has write permission if all of the following apply:

- AP[2] is 0 or DBM is 1 in the stage 1 descriptor.
- Where APTable is in use, APTable[1] is 0 for all levels of the translation table.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnIA, bit [31]

When FEAT_PAAuth is implemented:

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL2 or EL2&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnIA	Meaning
0b0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is enabled.

Note

This field controls the behavior of the AddPACIA and AuthIA pseudocode functions. Specifically, when the field is 1, AddPACIA returns a copy of a pointer to which a pointer authentication code has been added, and AuthIA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnIB, bit [30]**When FEAT_PAAuth is implemented:**

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL2 or EL2&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnIB	Meaning
0b0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is enabled.

Note

This field controls the behavior of the AddPACIB and AuthIB pseudocode functions. Specifically, when the field is 1, AddPACIB returns a copy of a pointer to which a pointer authentication code has been added, and AuthIB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LSMAOE, bit [29]**When FEAT_LSMAOC is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Load Multiple and Store Multiple Atomicity and Ordering Enable.

LSMAOE	Meaning
0b0	For all memory accesses at EL0, A32 and T32 Load Multiple and Store Multiple can have an interrupt taken during the sequence memory accesses, and the memory accesses are not required to be ordered.
0b1	The ordering and interrupt behavior of A32 and T32 Load Multiple and Store Multiple at EL0 is as defined for Armv8.0.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_LSMAOC is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

nTLSMD, bit [28]**When FEAT_LSMAOC is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory.

nTLSMD	Meaning
0b0	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are trapped and generate a stage 1 Alignment fault.
0b1	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are not trapped.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_LSMAOC is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

EnDA, bit [27]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL2 or EL2&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnDA	Meaning
0b0	Pointer authentication (using the APDAKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDAKey_EL1 key) of data addresses is enabled.

Note

This field controls the behavior of the AddPACDA and AuthDA pseudocode functions. Specifically, when the field is 1, AddPACDA returns a copy of a pointer to which a pointer authentication code has been added, and AuthDA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UCI, bit [26]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Traps execution of cache maintenance instructions at EL0 to EL2, from AArch64 state only. This applies to [DC CVAU](#), [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#), and [IC IVAU](#).

If FEAT_DPB2 is implemented, this trap also applies to [DC CVADP](#).

If FEAT_MTE is implemented, this trap also applies to [DC CIGVAC](#), [DC CIGDVAC](#), [DC CGVAC](#), [DC CGDVAC](#), [DC CGVAP](#), and [DC CGDVAP](#).

If FEAT_DPB2 and FEAT_MTE are implemented, this trap also applies to [DC CGVADP](#) and [DC CGDVADP](#).

UCI	Meaning
0b0	Any attempt to execute an instruction that this trap applies to at EL0 using AArch64 is trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

If the Point of Coherency is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean, or clean and invalidate instruction that operates by VA to the point of coherency can be trapped when the value of this control is 1.

If the Point of Unification is before any level of data cache, it is IMPLEMENTATION DEFINED whether the execution of any data or unified cache clean by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

If the Point of Unification is before any level of instruction cache, it is IMPLEMENTATION DEFINED whether the execution of any instruction cache invalidate by VA to the Point of Unification instruction can be trapped when the value of this control is 1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

EE, bit [25]

Endianness of data accesses at EL2, stage 1 translation table walks in the EL2 or EL2&0 translation regime, and stage 2 translation table walks in the EL1&0 translation regime.

EE	Meaning
0b0	Explicit data accesses at EL2, stage 1 translation table walks in the EL2 or EL2&0 translation regime, and stage 2 translation table walks in the EL1&0 translation regime are little-endian.
0b1	Explicit data accesses at EL2, stage 1 translation table walks in the EL2 or EL2&0 translation regime, and stage 2 translation table walks in the EL1&0 translation regime are big-endian.

If an implementation does not provide Big-endian support at Exception levels higher than EL0, this bit is RES0.

If an implementation does not provide Little-endian support at Exception levels higher than EL0, this bit is RES1.

The EE bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

E0E, bit [24]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Endianness of data accesses at EL0.

E0E	Meaning
0b0	Explicit data accesses at EL0 are little-endian.
0b1	Explicit data accesses at EL0 are big-endian.

If an implementation only supports Little-endian accesses at EL0, then this bit is RES0. This option is not permitted when SCTLR_EL1.EE is RES1.

If an implementation only supports Big-endian accesses at EL0, then this bit is RES1. This option is not permitted when SCTLR_EL1.EE is RES0.

This bit has no effect on the endianness of LDTR, LDTRH, LDTRSH, LDTRSW, STTR, and STTRH instructions executed at EL1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

SPAN, bit [23]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Set Privileged Access Never, on taking an exception to EL2.

SPAN	Meaning
0b0	PSTATE.PAN is set to 1 on taking an exception to EL2.
0b1	The value of PSTATE.PAN is left unchanged on taking an exception to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

EIS, bit [22]**When FEAT_ExS is implemented:**

Exception entry is a context synchronization event.

EIS	Meaning
0b0	The taking of an exception to EL2 is not a context synchronization event.
0b1	The taking of an exception to EL2 is a context synchronization event.

If SCTLR_EL2.EIS is set to 0b0:

- Indirect writes to [ESR_EL2](#), [FAR_EL2](#), [SPSR_EL2](#), [ELR_EL2](#), and [HPFAR_EL2](#) are synchronized on exception entry to EL2, so that a direct read of the register after exception entry sees the indirectly written value caused by the exception entry.
- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL2.EIS:

- Changes to the PSTATE information on entry to EL2.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores, and data processing instructions.
- Exit from Debug state.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

IESB, bit [21]**When FEAT_IESB is implemented:**

Implicit Error Synchronization event enable.

IESB	Meaning
0b0	Disabled.
0b1	An implicit error synchronization event is added: <ul style="list-style-type: none"> • At each exception taken to EL2. • Before the operational pseudocode of each ERET instruction executed at EL2.

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and its Effective value might be 0 or 1 regardless of the value of the field. If the Effective value of the field is 1, then an implicit error synchronization event is added after each DCPSX instruction taken to EL2 and before each DRPS instruction executed at EL2, in addition to the other cases where it is added.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TSCXT, bit [20]

When (FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented), HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Trap EL0 Access to the SCXTNUM_EL0 register, when EL0 is using AArch64.

TSCXT	Meaning
0b0	EL0 access to SCXTNUM_EL0 is not disabled by this mechanism.
0b1	EL0 access to SCXTNUM_EL0 is disabled, causing an exception to EL2, and the SCXTNUM_EL0 value is treated as 0.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_CSV2_2 is not implemented, FEAT_CSV2_1p2 is not implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Reserved, RES1.

When (FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented), HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

WXN, bit [19]

Write permission implies XN (Execute-never). For the EL2 or EL2&0 translation regime, this bit can force all memory regions that are writable to be treated as XN.

WXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable in the EL2 or EL2&0 translation regime is forced to XN for accesses from software executing at EL2.

This bit applies only when SCTLR_EL2.M bit is set.

The WXN bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

nTWE, bit [18]

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Traps execution of WFE instructions at EL0 to EL2, from both Execution states.

nTWE	Meaning
0b0	Any attempt to execute a WFE instruction at EL0 is trapped to EL2, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

In AArch32 state, the attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Traps execution of WFI instructions at EL0 to EL2, from both Execution states.

nTWI	Meaning
0b0	Any attempt to execute a WFI instruction at EL0 is trapped to EL2, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

In AArch32 state, the attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

UCT, bit [15]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Traps EL0 accesses to the [CTR_ELO](#) to EL2, from AArch64 state only.

UCT	Meaning
0b0	Accesses to the CTR_ELO from EL0 using AArch64 are trapped to EL2.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

DZE, bit [14]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

Traps execution of [DC ZVA](#) instructions at EL0 to EL2, from AArch64 state only.

If FEAT_MTE is implemented, this trap also applies to [DC GVA](#) and [DC GZVA](#).

DZE	Meaning
0b0	Any attempt to execute an instruction that this trap applies to at EL0 using AArch64 is trapped to EL2. Reading DCZID_ELO .DZP from EL0 returns 1, indicating that the instructions that this trap applies to are not supported.
0b1	This control does not cause any instructions to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

EnDB, bit [13]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL2 or EL2&0 translation regime.

For more information, see 'System register control of pointer authentication'.

EnDB	Meaning
0b0	Pointer authentication (using the APDBKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDBKey_EL1 key) of data addresses is enabled.

Note

This field controls the behavior of the AddPACDB and AuthDB pseudocode functions. Specifically, when the field is 1, AddPACDB returns a copy of a pointer to which a pointer authentication code has been added, and AuthDB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL2 and, when EL2 is enabled in the current Security state and [HCR_EL2](#).{E2H,TGE} == {1,1}, EL0.

I	Meaning
0b0	All instruction accesses to Normal memory from EL2 are Non-cacheable for all levels of instruction and unified cache. When EL2 is enabled in the current Security state and HCR_EL2 .{E2H, TGE} == {1, 1}, all instruction accesses to Normal memory from EL0 are Non-cacheable for all levels of instruction and unified cache. If SCTLR_EL2.M is 0, instruction accesses from stage 1 of the EL2 or EL2&0 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
0b1	This control has no effect on the Cacheability of instruction access to Normal memory from EL2 and, when EL2 is enabled in the current Security state and HCR_EL2 .{E2H, TGE} == {1, 1}, instruction access to Normal memory from EL0. If the value of SCTLR_EL2.M is 0, instruction accesses from stage 1 of the EL2 or EL2&0 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

This bit has no effect on the EL3 translation regime.

When EL2 is disabled in the current Security state or [HCR_EL2](#).{E2H,TGE} != {1,1}, this bit has no effect on the EL1&0 translation regime.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

EOS, bit [11]

When FEAT_ExS is implemented:

Exception exit is a context synchronization event.

EOS	Meaning
0b0	An exception return from EL2 is not a context synchronization event.
0b1	An exception return from EL2 is a context synchronization event.

If SCTLR_EL2.EOS is set to 0b0:

- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL2.EOS:

- The indirect write of the PSTATE and PC values from [SPSR_EL2](#) and [ELR_EL2](#) on exception return is synchronized.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores, and data processing instructions.
- Exit from Debug state.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

EnRCTX, bit [10]

When FEAT_SPECRES is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Enable EL0 ~~access~~Access to the following ~~System~~ instructions:

- ~~CFPRCTX, DVPRCTX and CPPRCTX~~ ~~AArch32 CFPRCTX, DVPRCTX and CPPRCTX~~ instructions.
- ~~If AArch64 FEAT_SPECRES2 is implemented, DVP RCT and CPP RCTX instructions.~~
~~COSPRCTX.~~
- ~~CFP RCTX, DVP RCTX and CPP RCTX~~ instructions.
- ~~If FEAT_SPECRES2 is implemented, COSP RCTX.~~

EnRCTX	Meaning
0b0	EL0 access to these instructions is disabled, and these instructions are trapped to EL1.
0b1	EL0 access to these instructions is enabled.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When FEAT_SPECRES is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

Bit [9]

Reserved, RES0.

SED, bit [8]

When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

SETEND instruction disable. Disables SETEND instructions at EL0 using AArch32.

SED	Meaning
0b0	SETEND instruction execution is enabled at EL0 using AArch32.
0b1	SETEND instructions are UNDEFINED at EL0 using AArch32.

If the implementation does not support mixed-endian operation at any Exception level, this bit is RES1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When EL0 can only use AArch64, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Reserved, RES1.

When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

ITD, bit [7]

When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

IT Disable. Disables some uses of IT instructions at EL0 using AArch32.

ITD	Meaning
0b0	All IT instruction functionality is enabled at EL0 using AArch32.
0b1	<p>Any attempt at EL0 using AArch32 to execute any of the following is UNDEFINED:</p> <ul style="list-style-type: none"> All encodings of the IT instruction with hw1[3:0] != 1000. All encodings of the subsequent instruction with the following values for hw1: <ul style="list-style-type: none"> 0b11xxxxxxxxxxxx: All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM. 0b1011xxxxxxxxxxxx: All instructions in 'Miscellaneous 16-bit instructions'. 0b10100xxxxxxxxxxx: ADD Rd, PC, #imm 0b01001xxxxxxxxxxx: LDR Rd, [PC, #imm] 0b0100x1xxx1111xxx: ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC. 0b010001xx1xxxx111: ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers UNPREDICTABLE cases with BLX Rn. <p>These instructions are always UNDEFINED, regardless of whether they would pass or fail the condition code check that applies to them as a result of being in an IT block.</p> <p>It is IMPLEMENTATION DEFINED whether the IT instruction is treated as:</p> <ul style="list-style-type: none"> A 16-bit instruction, that can only be followed by another 16-bit instruction. The first half of a 32-bit instruction. <p>This means that, for the situations that are UNDEFINED, either the second 16-bit instruction or the 32-bit instruction is UNDEFINED.</p> <p>An implementation might vary dynamically as to whether IT is treated as a 16-bit instruction or the first half of a 32-bit instruction.</p>

If an instruction in an active IT block that would be disabled by this field sets this field to 1 then behavior is CONSTRAINED UNPREDICTABLE. For more information see 'Changes to an ITD control by an instruction in an IT block'.

ITD is optional, but if it is implemented in the SCTLR_EL2 then it must also be implemented in the [SCTLR_EL1](#), [HSCTLR](#), and [SCTLR](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When an implementation does not implement ITD, access to this field is **RAZ/WI**.

When EL0 can only use AArch64, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Reserved, RES1.

When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES0.

nAA, bit [6]**When FEAT_LSE2 is implemented:**

Non-aligned access. This bit controls generation of Alignment faults under certain conditions at EL2, and, when EL2 is enabled in the current Security state and [HCR_EL2](#).{E2H, TGE} == {1, 1}, EL0.

The following instructions generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH.
- STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH

The following instructions generate an Alignment fault if all bytes being accessed for a single register are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDIAPP, LDAPR, STILP, and STLR.
- If Advanced SIMD and floating-point instructions are implemented, LDDAPUR, LDAP1, STLUR, and STL1.

nAA	Meaning
0b0	Unaligned accesses by the specified instructions generate an Alignment fault.
0b1	Unaligned accesses by the specified instructions do not generate an Alignment fault.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CP15BEN, bit [5]**When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

System instruction memory barrier enable. Enables accesses to the DMB, DSB, and ISB System instructions in the (coproc==0b1111) encoding space from EL0:

CP15BEN	Meaning
0b0	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
0b1	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

CP15BEN is optional, but if it is implemented in the SCTLR_EL2 then it must also be implemented in the [SCTLR_EL1](#), [HSCTLR](#), and [SCTLR](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

When EL0 can only use AArch64, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Reserved, RES0.

When EL0 is capable of using AArch32, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 0:

IGNORED.

Otherwise:

Reserved, RES1.

SA0, bit [4]**When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:**

SP Alignment check enable for EL0. When set to 1, if a load or store instruction executed at EL0 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see 'SP alignment checking'.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

SA, bit [3]

SP Alignment check enable. When set to 1, if a load or store instruction executed at EL2 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see 'SP alignment checking'.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

C, bit [2]

Data access Cacheability control, for accesses at EL2 and, when EL2 is enabled in the current Security state and [HCR_EL2](#).{E2H, TGE} == {1, 1}, EL0

C	Meaning
0b0	<p>The following are Non-cacheable for all levels of data and unified cache:</p> <ul style="list-style-type: none"> • Data accesses to Normal memory from EL2. • When HCR_EL2.{E2H, TGE} != {1, 1}, Normal memory accesses to the EL2 translation tables. • When EL2 is enabled in the current Security state and HCR_EL2.{E2H, TGE} == {1, 1}: <ul style="list-style-type: none"> ◦ Data accesses to Normal memory from EL0. ◦ Normal memory accesses to the EL2&0 translation tables.
0b1	<p>This control has no effect on the Cacheability of:</p> <ul style="list-style-type: none"> • Data access to Normal memory from EL2. • When HCR_EL2.{E2H, TGE} != {1, 1}, Normal memory accesses to the EL2 translation tables. • When EL2 is enabled in the current Security state and HCR_EL2.{E2H, TGE} == {1, 1}: <ul style="list-style-type: none"> ◦ Data accesses to Normal memory from EL0. ◦ Normal memory accesses to the EL2&0 translation tables.

This bit has no effect on the EL3 translation regime.

When EL2 is disabled in the current Security state or [HCR_EL2](#).{E2H, TGE} != {1, 1}, this bit has no effect on the EL1&0 translation regime.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL2 and, when EL2 is enabled in the current Security state and [HCR_EL2](#).{E2H, TGE} == {1, 1}, EL0.

A	Meaning
0b0	Alignment fault checking disabled when executing at EL2. When EL2 is enabled in the current Security state and HCR_EL2 .{E2H, TGE} == {1, 1}, alignment fault checking disabled when executing at EL0. Instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, do not check that the address being accessed is aligned to the size of the data element(s) being accessed.
0b1	Alignment fault checking enabled when executing at EL2. When EL2 is enabled in the current Security state and HCR_EL2 .{E2H, TGE} == {1, 1}, alignment fault checking enabled when executing at EL0. All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have an alignment check regardless of the value of the A bit.

If FEAT_MOPS is implemented, SETG* instructions have an alignment check regardless of the value of the A bit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

M, bit [0]

MMU enable for EL2 or EL2&0 stage 1 address translation.

M	Meaning
0b0	When HCR_EL2 .{E2H, TGE} != {1, 1}, EL2 stage 1 address translation disabled. When HCR_EL2 .{E2H, TGE} == {1, 1}, EL2&0 stage 1 address translation disabled. See the SCTLR_EL2.I field for the behavior of instruction accesses to Normal memory.
0b1	When HCR_EL2 .{E2H, TGE} != {1, 1}, EL2 stage 1 address translation enabled. When HCR_EL2 .{E2H, TGE} == {1, 1}, EL2&0 stage 1 address translation enabled.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2, this field resets to 0.

Accessing SCTLR_EL2

When [HCR_EL2](#).E2H is 1, without explicit synchronization, access from EL2 using the mnemonic SCTLR_EL2 or SCTLR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SCTLR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR_EL2;

```

MSR SCTLR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SCTLR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR_EL2 = X[t, 64];

```

MRS <Xt>, SCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x110];
    else
        X[t, 64] = SCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR_EL2;
    else
        X[t, 64] = SCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR_EL1;

```

MSR SCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.SCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x110] = X[t, 64];
    else
        SCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SCTLR_EL2 = X[t, 64];
    else
        SCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SCTLR_EL3, System Control Register (EL3)

The SCTLR_EL3 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to SCTLR_EL3 are UNDEFINED.

Attributes

SCTLR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44
RES0	SPINTMASK	NMI	RES0	TCSOTME	RES0				TME	TMT	RES0	TMT	DSSBS	RES0				ATA	DSSBS
EnIA	EnIB	RES1	EnDA	RES0	EE	RES0	RES1	EIS	IESB	RES0	WXN	RES1	RES0	RES1	RES0	EnDB			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12

Bit [63]

Reserved, RES0.

SPINTMASK, bit [62] When FEAT_NMI is implemented:

SP Interrupt Mask enable. When SCTLR_EL3.NMI is 1, controls whether PSTATE.SP acts as an interrupt mask, and controls the value of PSTATE.ALLINT on taking an exception to EL3.

SPINTMASK	Meaning
0b0	Does not cause PSTATE.SP to mask interrupts. PSTATE.ALLINT is set to 1 on taking an exception to EL3.
0b1	When PSTATE.SP is 1 and execution is at EL3, an IRQ or FIQ interrupt that is targeted to EL3 is masked regardless of any denotion of Superpriority. PSTATE.ALLINT is set to 0 on taking an exception to EL3.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NMI, bit [61]**When FEAT_NMI is implemented:**

Non-maskable Interrupt enable.

NMI	Meaning
0b0	This control does not affect interrupt masking behavior.
0b1	This control enables all of the following: <ul style="list-style-type: none"> The use of the PSTATE.ALLINT interrupt mask. IRQ and FIQ interrupts to have Superpriority as an additional attribute. PSTATE.SP to be used as an interrupt mask.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

BitBits [60:54]

Reserved, RES0.

TCSO, bit [59]**When FEAT_MTE_STORE_ONLY is implemented:**

Tag Checking Store Only.

TCSO	Meaning
0b0	This field has no effect on Tag checking.
0b1	Load instructions executed in EL3 are Tag Unchecked.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [58:54]

Reserved, RES0.

TME, bit [53]**When FEAT_TME is implemented:**

Enables the Transactional Memory Extension at EL3.

TME	Meaning
0b0	Any attempt to execute a TSTART instruction at EL3 is trapped, unless HCR_EL2.TME or SCR_EL3.TME causes TSTART instructions to be UNDEFINED at EL3.
0b1	This control does not cause any TSTART instruction to be trapped.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [52]

Reserved, RES0.

TMT, bit [51]**When FEAT_TME is implemented:**

Forces a trivial implementation of the Transactional Memory Extension at EL3.

TMT	Meaning
0b0	This control does not cause any TSTART instruction to fail.
0b1	When the TSTART instruction is executed at EL3, the transaction fails with a TRIVIAL failure cause.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [50:45]

Reserved, RES0.

DSSBS, bit [44]**When FEAT_SSBS is implemented:**

Default PSTATE.SSBS value on Exception Entry.

DSSBS	Meaning
0b0	PSTATE.SSBS is set to 0 on an exception to EL3.
0b1	PSTATE.SSBS is set to 1 on an exception to EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Otherwise:

Reserved, RES0.

ATA, bit [43]**When FEAT_MTE2 is implemented:**

Allocation Tag Access in EL3.

Controls access to Allocation Tags and Tag Check operations in EL3.

ATA	Meaning
0b0	Access to Allocation Tags is prevented at EL3. Memory accesses at EL3 are not subject to a Tag Check operation.
0b1	This control does not prevent access to Allocation Tags at EL3. Tag Checked memory accesses at EL3 are subject to a Tag Check operation. The Tag Check operation depends on the type of tag at the memory being accessed: <ul style="list-style-type: none"> For Allocation Tagged memory, an Allocation Tag Check operation. If FEAT_MTE_CANONICAL_TAGS is implemented, for Canonically Tagged memory, a Canonical Tag Check operation.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [42]

Reserved, RES0.

TCF, bits [41:40]

When FEAT_MTE2 is implemented:

Tag Check Fault in EL3. Controls the effect of Tag Check Faults due to Loads and Stores in EL3.

If FEAT_MTE3 is not implemented, the value 0b11 is reserved.

TCF	Meaning	Applies when
0b00	Tag Check Faults have no effect on the PE.	
0b01	Tag Check Faults cause a synchronous exception.	
0b10	Tag Check Faults are asynchronously accumulated.	
0b11	Tag Check Faults cause a synchronous exception on reads, and are asynchronously accumulated on writes.	When FEAT_MTE3 is implemented

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [39:38]

Reserved, RES0.

ITFSB, bit [37]**When FEAT_MTE2 is implemented:**

When synchronous exceptions are not being generated by Tag Check Faults, this field controls whether on exception entry into EL3, all Tag Check Faults due to instructions executed before exception entry, that are reported asynchronously, are synchronized into [TFSRE0_EL1](#) and TFSR_ELx registers.

ITFSB	Meaning
0b0	Tag Check Faults are not synchronized on entry to EL3.
0b1	Tag Check Faults are synchronized on entry to EL3.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT, bit [36]**When FEAT_BTI is implemented:**

PAC Branch Type compatibility at EL3.

BT	Meaning
0b0	When the PE is executing at EL3, PACIASP and PACIBSP are compatible with PSTATE.BTYPE == 0b11.
0b1	When the PE is executing at EL3, PACIASP and PACIBSP are not compatible with PSTATE.BTYPE == 0b11.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [35:32]

Reserved, RES0.

EnIA, bit [31]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL3 translation regime.

Possible values of this bit are:

EnIA	Meaning
0b0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is enabled.

For more information, see 'System register control of pointer authentication'.

Note

This field controls the behavior of the AddPACIA and AuthIA pseudocode functions. Specifically, when the field is 1, AddPACIA returns a copy of a pointer to which a pointer authentication code has been added, and AuthIA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EnIB, bit [30]

When FEAT_PAAuth is implemented:

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL3 translation regime.

Possible values of this bit are:

EnIB	Meaning
0b0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
0b1	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is enabled.

For more information, see 'System register control of pointer authentication'.

Note

This field controls the behavior of the AddPACIB and AuthIB pseudocode functions. Specifically, when the field is 1, AddPACIB returns a copy of a pointer to which a pointer authentication code has been added, and AuthIB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:28]

Reserved, RES1.

EnDA, bit [27]

When FEAT_PAAuth is implemented:

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL3 translation regime.

EnDA	Meaning
0b0	Pointer authentication (using the APDAKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDAKey_EL1 key) of data addresses is enabled.

For more information, see 'System register control of pointer authentication'.

Note

This field controls the behavior of the AddPACDA and AuthDA pseudocode functions. Specifically, when the field is 1, AddPACDA returns a copy of a pointer to which a pointer authentication code has been added, and AuthDA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [26]

Reserved, RES0.

EE, bit [25]

Endianness of data accesses at EL3, and stage 1 translation table walks in the EL3 translation regime.

EE	Meaning
0b0	Explicit data accesses at EL3, and stage 1 translation table walks in the EL3 translation regime are little-endian.
0b1	Explicit data accesses at EL3, and stage 1 translation table walks in the EL3 translation regime are big-endian.

If an implementation does not provide Big-endian support at Exception levels higher than EL0, this bit is RES0.

If an implementation does not provide Little-endian support at Exception levels higher than EL0, this bit is RES1.

The EE bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Bit [24]

Reserved, RES0.

Bit [23]

Reserved, RES1.

EIS, bit [22]

When FEAT_ExS is implemented:

Exception Entry is Context Synchronizing.

EIS	Meaning
0b0	The taking of an exception to EL3 is not a context synchronizing event.
0b1	The taking of an exception to EL3 is a context synchronizing event.

If SCTLR_EL3.EIS is set to 0b0:

- Indirect writes to [ESR_EL3](#), [FAR_EL3](#), [SPSR_EL3](#), [ELR_EL3](#) are synchronized on exception entry to EL3, so that a direct read of the register after exception entry sees the indirectly written value caused by the exception entry.
- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL3.EIS:

- Changes to the PSTATE information on entry to EL3.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores and data processing instructions.
- Debug state exit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

IESB, bit [21]

When FEAT_IESB is implemented:

Implicit Error Synchronization event enable.

IESB	Meaning
0b0	Disabled.
0b1	An implicit error synchronization event is added: <ul style="list-style-type: none"> • At each exception taken to EL3. • Before the operational pseudocode of each ERET instruction executed at EL3.

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and its Effective value might be 0 or 1 regardless of the value of the field and, if implemented, [SCR_EL3.NMEA](#). If the Effective value of the field is 1, then an implicit error synchronization event is added after each DCPSX instruction taken to EL3 and before each DRPS instruction executed at EL3, in addition to the other cases where it is added.

When FEAT_DoubleFault is implemented, the PE is in Non-debug state, and the Effective value of [SCR_EL3.NMEA](#) is 1, this field is ignored and its Effective value is 1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [20]

Reserved, RES0.

WXN, bit [19]

Write permission implies XN (Execute-never). For the EL3 translation regime, this bit can force all memory regions that are writable to be treated as XN.

WXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable in the EL3 translation regime is forced to XN for accesses from software executing at EL3.

This bit applies only when SCTLR_EL3.M bit is set.

The WXN bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Bit [18]

Reserved, RES1.

Bit [17]

Reserved, RES0.

Bit [16]

Reserved, RES1.

Bits [15:14]

Reserved, RES0.

EnDB, bit [13]**When FEAT_PAuth is implemented:**

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL3 translation regime.

EnDB	Meaning
0b0	Pointer authentication (using the APDBKey_EL1 key) of data addresses is not enabled.
0b1	Pointer authentication (using the APDBKey_EL1 key) of data addresses is enabled.

For more information, see 'System register control of pointer authentication'.

Note

This field controls the behavior of the AddPACDB and AuthDB pseudocode functions. Specifically, when the field is 1, AddPACDB returns a copy of a pointer to which a pointer authentication code has been added, and AuthDB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL3:

I	Meaning
0b0	All instruction access to Normal memory from EL3 are Non-cacheable for all levels of instruction and unified cache. If the value of SCTLR_EL3.M is 0, instruction accesses from stage 1 of the EL3 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
0b1	This control has no effect on the Cacheability of instruction access to Normal memory from EL3. If the value of SCTLR_EL3.M is 0, instruction accesses from stage 1 of the EL3 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

This bit has no effect on the EL1&0, EL2, or EL2&0 translation regimes.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to 0.

EOS, bit [11]**When FEAT_ExS is implemented:**

Exception Exit is Context Synchronizing.

EOS	Meaning
0b0	An exception return from EL3 is not a context synchronizing event
0b1	An exception return from EL3 is a context synchronizing event

If SCTLR_EL3.EOS is set to 0b0:

- Memory transactions, including instruction fetches, from an Exception level always use the translation resources associated with that translation regime.
- Exception Catch debug events are synchronous debug events.
- DCPS* and DRPS instructions are context synchronization events.

The following are not affected by the value of SCTLR_EL3.EOS:

- The indirect write of the PSTATE and PC values from [SPSR_EL3](#) and [ELR_EL3](#) on exception return is synchronized.
- If the PE enters Debug state before the first instruction after an Exception return from EL3 to Non-secure state, any pending Halting debug event completes execution.
- The GIC behavior that allocates interrupts to FIQ or IRQ changes simultaneously with leaving the EL3 Exception level.
- Behavior of accessing the banked copies of the stack pointer using the SP register name for loads, stores and data processing instructions.
- Exit from Debug state.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

Bits [10:7]

Reserved, RES0.

nAA, bit [6]**When FEAT_LSE2 is implemented:**

Non-aligned access. This bit controls generation of Alignment faults at EL3 under certain conditions. The following instructions generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH.
- STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH

The following instructions generate an Alignment fault if all bytes being accessed for a single register are not within a single 16-byte quantity, aligned to 16 bytes for access:

- LDIAPP, LDAPR, STILP, and STLR.
- If Advanced SIMD and floating-point instructions are implemented, LDDAPUR, LDAP1, STLUR, and STL1.

nAA	Meaning
0b0	Unaligned accesses by the specified instructions generate an Alignment fault.
0b1	Unaligned accesses by the specified instructions do not generate an Alignment fault.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:4]

Reserved, RES1.

SA, bit [3]

SP Alignment check enable. When set to 1, if a load or store instruction executed at EL3 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then a SP alignment fault exception is generated. For more information, see 'SP alignment checking'.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

C, bit [2]

Cacheability control, for data accesses.

C	Meaning
0b0	All data access to Normal memory from EL3, and all Normal memory accesses to the EL3 translation tables, are Non-cacheable for all levels of data and unified cache.
0b1	This control has no effect on the Cacheability of: <ul style="list-style-type: none"> Data access to Normal memory from EL3. Normal memory accesses to the EL3 translation tables.

This bit has no effect on the EL1&0, EL2, or EL2&0 translation regimes.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL3.

A	Meaning
0b0	Alignment fault checking disabled when executing at EL3. Instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, do not check that the address being accessed is aligned to the size of the data element(s) being accessed.
0b1	Alignment fault checking enabled when executing at EL3. All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have an alignment check regardless of the value of the A bit.

If FEAT_MOPS is implemented, SETG* instructions have an alignment check regardless of the value of the A bit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to an architecturally UNKNOWN value.

M, bit [0]

MMU enable for EL3 stage 1 address translation. Possible values of this bit are:

M	Meaning
0b0	EL3 stage 1 address translation disabled. See the SCTLR_EL3.I field for the behavior of instruction accesses to Normal memory.
0b1	EL3 stage 1 address translation enabled.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL3, this field resets to 0.

Accessing SCTLR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCTLR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR_EL3;
```

MSR SCTLR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SCTLR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SCXTNUM_EL0, EL0 Read/Write Software Context Number

The SCXTNUM_EL0 characteristics are:

Purpose

Provides a number that can be used to separate out different context numbers with the EL0 exception level, for the purpose of protecting against side-channels using branch prediction and similar resources.

Configuration

This register is present only when FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented. Otherwise, direct accesses to SCXTNUM_EL0 are UNDEFINED.

Attributes

SCXTNUM_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Software Context Number																															
Software Context Number																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Software Context Number. A number to identify the context within the EL0 exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SCXTNUM_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCXTNUM_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.TSCXT == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.EnSCXT == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.SCXTNUM_EL0 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.TSCXT == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = SCXTNUM_EL0;
        elsif PSTATE.EL == EL1 then
            if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
                UNDEFINED;
            elsif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.SCXTNUM_EL0 == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.SystemAccessTrap(EL3, 0x18);
                else
                    X[t, 64] = SCXTNUM_EL0;
            elsif PSTATE.EL == EL2 then
                if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
                    UNDEFINED;
                elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
                    if Halted() && EDSCR.SDD == '1' then
                        UNDEFINED;
                    else
                        AArch64.SystemAccessTrap(EL3, 0x18);
                    else
                        X[t, 64] = SCXTNUM_EL0;
            elsif PSTATE.EL == EL3 then
                X[t, 64] = SCXTNUM_EL0;

```

MSR SCXTNUM_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elseif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.TSCXT == '1' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.SCXTNUM_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.TSCXT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCXTNUM_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elseif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCXTNUM_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCXTNUM_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elseif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCXTNUM_EL0 = X[t, 64];
elseif PSTATE.EL == EL3 then
    SCXTNUM_EL0 = X[t, 64];

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SCXTNUM_EL1, EL1 Read/Write Software Context Number

The SCXTNUM_EL1 characteristics are:

Purpose

Provides a number that can be used to separate out different context numbers with the EL1 exception level, for the purpose of protecting against side-channels using branch prediction and similar resources.

Configuration

This register is present only when FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented. Otherwise, direct accesses to SCXTNUM_EL1 are UNDEFINED.

Attributes

SCXTNUM_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Software Context Number																															
Software Context Number																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Software Context Number. A number to identify the context within the EL1 exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SCXTNUM_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCXTNUM_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.SCXTNUM_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            X[t, 64] = NVMem[0x188];
        else
            X[t, 64] = SCXTNUM_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            X[t, 64] = SCXTNUM_EL2;
        else
            X[t, 64] = SCXTNUM_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = SCXTNUM_EL1;

```

MSR SCXTNUM_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCXTNUM_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x188] = X[t, 64];
        else
            SCXTNUM_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            SCXTNUM_EL2 = X[t, 64];
        else
            SCXTNUM_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        SCXTNUM_EL1 = X[t, 64];

```

MRS <Xt>, SCXTNUM_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1101	0b0000	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x188];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = SCXTNUM_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = SCXTNUM_EL1;
    else
        UNDEFINED;

```

MSR SCXTNUM_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x188] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SCXTNUM_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SCXTNUM_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SCXTNUM_EL2, EL2 Read/Write Software Context Number

The SCXTNUM_EL2 characteristics are:

Purpose

Provides a number that can be used to separate out different context numbers with the EL2 exception level, for the purpose of protecting against side-channels using branch prediction and similar resources.

Configuration

This register is present only when FEAT_CSV2_2 is implemented or FEAT_CSV2_1p2 is implemented. Otherwise, direct accesses to SCXTNUM_EL2 are UNDEFINED.

If EL2 is not implemented, this register is RES0 from EL3.

Attributes

SCXTNUM_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Software Context Number																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Software Context Number. A number to identify the context within the EL2 exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SCXTNUM_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic SCXTNUM_EL2 or SCXTNUM_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SCXTNUM_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SCXTNUM_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCXTNUM_EL2;

```

MSR SCXTNUM_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCXTNUM_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCXTNUM_EL2 = X[t, 64];

```

MRS <Xt>, SCXTNUM_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.SCXTNUM_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            X[t, 64] = NVMem[0x188];
        else
            X[t, 64] = SCXTNUM_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            X[t, 64] = SCXTNUM_EL2;
        else
            X[t, 64] = SCXTNUM_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = SCXTNUM_EL1;

```

MSR SCXTNUM_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.EnSCXT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.SCXTNUM_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x188] = X[t, 64];
        else
            SCXTNUM_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnSCXT == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnSCXT == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            SCXTNUM_EL2 = X[t, 64];
        else
            SCXTNUM_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        SCXTNUM_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Otherwise:

Reserved, RES0.

EZT0, Bits bit [30:9]
When FEAT_SME2 is implemented:

Traps execution at EL1 and EL0 of the LDR, LUT12, LUT14, MOVT, STR, and ZERO instructions that access the ZT0 register to EL1, or to EL2 when EL2 is implemented and enabled in the current Security state and HCR_EL2.TGE is 1.

The exception is reported using ESR_EL1.EC or ESR_EL2.EC value 0x1D, with an ISS code of 0x0000004, at a lower priority than a trap due to PSTATE.SM or PSTATE.ZA.

EZT0	Meaning
0b0	This control causes execution of these instructions at EL1 and EL0 to be trapped.
0b1	This control does not cause execution of any instruction to be trapped.

Changes to this field only affect whether instructions that access ZT0 are trapped. They do not affect the contents of ZT0, which remain valid so long as PSTATE.ZA is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Streaming SVE vector length (SVL) at EL1 of (LEN+1)*128 bits. This field also defines the Effective Streaming SVE vector length at EL0 when EL2 is not implemented, or EL2 is not enabled in the current Security state, or HCR_EL2.{E2H,TGE} is not {1,1}.

The Streaming SVE vector length can be any power of two from 128 bits to 2048 bits inclusive. An implementation can support any subset of the architecturally permitted lengths.

When the PE is in Streaming SVE mode, the Effective SVE vector length (VL) is equal to SVL.

When FEAT_SVE is implemented, and the PE is not in Streaming SVE mode, VL is equal to the Effective Non-streaming SVE vector length. See ZCR_EL1.

For all purposes other than returning the result of a direct read of SMCR_EL1, the PE selects the Effective Streaming SVE vector length by performing checks in the following order:

- If the requested length is less than the minimum implemented Streaming SVE vector length, then the Effective length is the minimum implemented Streaming SVE vector length.
- If EL2 is implemented and enabled in the current Security state, and the requested length is greater than the Effective length at EL2, then the Effective length at EL2 is used.

3. If EL3 is implemented and the requested length is greater than the Effective length at EL3, then the Effective length at EL3 is used.
4. Otherwise, the Effective length is the highest supported Streaming SVE vector length that is less than or equal to the requested length.

An indirect read of SMCR_EL1.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SMCR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic SMCR_EL1 or SMCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SMCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x1F0];
    else
        X[t, 64] = SMCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = SMCR_EL2;
    else
        X[t, 64] = SMCR_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL1;

```

MSR SMCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        endif
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x1F0] = X[t, 64];
    else
        SMCR_EL1 = X[t, 64];
endif
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        endif
    elsif HCR_EL2.E2H == '1' then
        SMCR_EL2 = X[t, 64];
    else
        SMCR_EL1 = X[t, 64];
endif
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL1 = X[t, 64];
    endif
endif

```

MRS <Xt>, SMCR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x1F0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
            UNDEFINED;
        elsif CPTR_EL3.SMEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            X[t, 64] = SMCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            X[t, 64] = SMCR_EL1;
    else
        UNDEFINED;

```

MSR SMCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x1F0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
            UNDEFINED;
        elsif CPTR_EL2.SMEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            SMCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            SMCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5809: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Otherwise:

Reserved, RES0.

EZT0, Bits bit [30:9]
When FEAT_SME2 is implemented:

Traps execution at EL2, EL1, and EL0 of the LDR, LUTi2, LUTi4, MOVt, STR, and ZERO instructions that access the ZT0 register to EL2, when EL2 is enabled in the current Security state.

The exception is reported using [ESR_EL2](#).EC value 0x1D, with an ISS code of 0x0000004, at a lower priority than a trap due to PSTATE.SM or PSTATE.ZA.

EZT0	Meaning
0b0	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b1	This control does not cause execution of any instruction to be trapped.

Changes to this field only affect whether instructions that access ZT0 are trapped. They do not affect the contents of ZT0, which remain valid so long as PSTATE.ZA is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Streaming SVE vector length (SVL) at EL2 of (LEN+1)*128 bits. This field also defines the Effective Streaming SVE vector length at EL0 when EL2 is implemented and enabled in the current Security state, and HCR_EL2.{E2H,TGE} is {1,1}.

The Streaming SVE vector length can be any power of two from 128 bits to 2048 bits inclusive. An implementation can support any subset of the architecturally permitted lengths.

When the PE is in Streaming SVE mode, the Effective SVE vector length (VL) is equal to SVL.

When FEAT_SVE is implemented, and the PE is not in Streaming SVE mode, VL is equal to the Effective Non-streaming SVE vector length. See [ZCR_EL2](#).

For all purposes other than returning the result of a direct read of SMCR_EL2, the PE selects the Effective Streaming SVE vector length by performing checks in the following order:

- If the requested length is less than the minimum implemented Streaming SVE vector length, then the Effective length is the minimum implemented Streaming SVE vector length.
- If EL3 is implemented and the requested length is greater than the Effective length at EL3, then the Effective length at EL3 is used.

- Otherwise, the Effective length is the highest supported Streaming SVE vector length that is less than or equal to the requested length.

An indirect read of SMCR_EL2.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SMCR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic SMCR_EL2 or SMCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SMCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL2;

```

MSR SMCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b110


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL2 = X[t, 64];

```

MRS <Xt>, SMCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        endif
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x1F0];
    else
        X[t, 64] = SMCR_EL1;
    endif
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        endif
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = SMCR_EL2;
    else
        X[t, 64] = SMCR_EL1;
    endif
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL1;
    endif
endif

```

MSR SMCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x1F0] = X[t, 64];
        else
            SMCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
            UNDEFINED;
        elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x1D);
            elsif HCR_EL2.E2H == '1' then
                SMCR_EL2 = X[t, 64];
            else
                SMCR_EL1 = X[t, 64];
        elsif PSTATE.EL == EL3 then
            if CPTR_EL3.ESM == '0' then
                AArch64.SystemAccessTrap(EL3, 0x1D);
            else
                SMCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The SMCR_EL3 characteristics are:

Purpose

This register controls aspects of Streaming SVE that are visible at all Exception levels.

Configuration

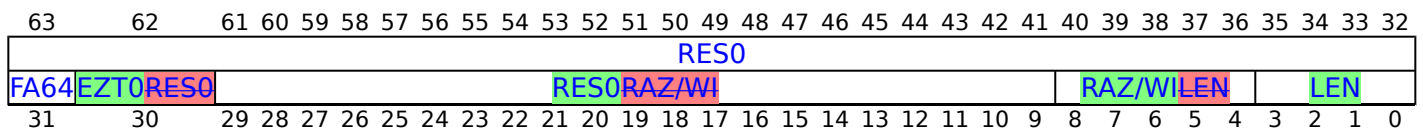
This register is present only when FEAT_SME is implemented and EL3 is implemented. Otherwise, direct accesses to SMCR_EL3 are UNDEFINED.

This register has no effect if the PE is not in Streaming SVE mode.

Attributes

SMCR_EL3 is a 64-bit register.

Field descriptions

**Bits [63:32]**

Reserved, RES0.

FA64, bit [31]

When FEAT_SME_FA64 is implemented:

Controls whether execution of an A64 instruction is considered legal when the PE is in Streaming SVE mode.

FA64	Meaning
0b0	This control does not cause any instruction to be treated as legal in Streaming SVE mode.
0b1	This control causes all implemented A64 instructions to be treated as legal in Streaming SVE mode at EL3.

Arm recommends that portable SME software should not rely on this optional feature, and that operating systems should provide a means to test for compliance with this recommendation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EZT0, Bits bit [30:9]
When FEAT_SME2 is implemented:

Traps execution at all Exception levels of the LDR, LUTi2, LUTi4, MOVt, STR, and ZERO instructions that access the ZT0 register to EL3.

The exception is reported using [ESR_EL3](#).EC value 0x1D, with an ISS code of 0x0000004, at a lower priority than a trap due to PSTATE.SM or PSTATE.ZA.

EZT0	Meaning
0b0	This control causes execution of these instructions at all Exception levels to be trapped.
0b1	This control does not cause execution of any instruction to be trapped.

Changes to this field only affect whether instructions that access ZT0 are trapped. They do not affect the contents of ZT0, which remain valid so long as PSTATE.ZA is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [29:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Streaming SVE vector length (SVL) at EL3 of (LEN+1)*128 bits.

The Streaming SVE vector length can be any power of two from 128 bits to 2048 bits inclusive. An implementation can support any subset of the architecturally permitted lengths.

When the PE is in Streaming SVE mode, the Effective SVE vector length (VL) is equal to SVL.

When FEAT_SVE is implemented, and the PE is not in Streaming SVE mode, VL is equal to the Effective Non-streaming SVE vector length. See [ZCR_EL3](#).

For all purposes other than returning the result of a direct read of SMCR_EL3, the PE selects the Effective Streaming SVE vector length by performing checks in the following order:

- If the requested length is less than the minimum implemented Streaming SVE vector length, then the Effective length is the minimum implemented Streaming SVE vector length.
- Otherwise, the Effective length is the highest supported Streaming SVE vector length that is less than or equal to the requested length.

An indirect read of SMCR_EL3.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

(old)

htmldiff from-

(new)

SMPRI_EL1, Streaming Mode Priority Register

The SMPRI_EL1 characteristics are:

Purpose

Configures the streaming execution priority for instructions executed on a shared Streaming Mode Compute Unit (SMCU) when the PE is in Streaming SVE mode at any Exception Level.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SMPRI_EL1 are UNDEFINED.

When [SMIDR_EL1](#).SMPS is '0', this register is RES0.

Attributes

SMPRI_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																RES0																					
																RES0														Priority							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

Bits [63:4]

Reserved, RES0.

Priority, bits [3:0]

Streaming execution priority value.

Either this value is used directly, or it is mapped into an effective priority value using [SMPRMAP_EL2](#).

This value is used directly when any of the following are true:

- The current Exception level is EL3 or EL2.
- The current Exception level is EL1 or EL0, if EL2 is implemented and enabled in the current Security state and [HCRX_EL2](#).SMPME is '0'.
- The current Exception level is EL1 or EL0, if EL2 is either not implemented or not enabled in the current Security state.

The precise meaning and behavior of each streaming execution priority value is IMPLEMENTATION DEFINED.

In an implementation that shares execution resources between PEs, higher priority values are allocated more processing resource than other PEs configured with lower priority values in the same Priority domain.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SMPRI_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SMPRI_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nSMPRI_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SMPRI_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SMPRI_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SMPRI_EL1;

```

MSR SMPRI_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b100


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nSMPRI_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SMPRI_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SMPRI_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SMPRI_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMACCESSR_EL1, System Performance Monitors Access Register (EL1)

The SPMACCESSR_EL1 characteristics are:

Purpose

Controls access to System PMUs from EL0.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMACCESSR_EL1 are UNDEFINED.

Attributes

SPMACCESSR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

P<m>, bits [2m+1:2m], for m = 31 to 0

System PMU <m> access. Controls access to System PMU <m>.

P<m>	Meaning
0b00	MRS read and MSR write System register accesses to System PMU <m> at EL0 are trapped to EL1, unless the instruction generates a higher priority exception.
0b01	MSR write System register accesses to System PMU <m> at EL0 are trapped to EL1, unless the instruction generates a higher priority exception.
0b11	This control does not cause any instructions to be trapped.

All other values are reserved.

The registers trapped by this control are:

AArch64: [SPMCNTENCLR_EL0](#), [SPMCNTENSET_EL0](#), [SPMCR_EL0](#), [SPMEVCNTR<n>_EL0](#), [SPMEVFILT2R<n>_EL0](#), [SPMEVFILTR<n>_EL0](#), [SPMEVTYPER<n>_EL0](#), [SPMOVSLR_EL0](#), and [SPMOVSSET_EL0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMACCESSR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMACCESSR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E8];
    else
        X[t, 64] = SPMACCESSR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SPMACCESSR_EL2;
    else
        X[t, 64] = SPMACCESSR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMACCESSR_EL1;

```

MSR SPMACCESSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E8] = X[t, 64];
    else
        SPMACCESSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPMACCESSR_EL2 = X[t, 64];
    else
        SPMACCESSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMACCESSR_EL1 = X[t, 64];

```

MRS <Xt>, SPMACCESSR_EL12

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x8E8];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SPMACCESSR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = SPMACCESSR_EL1;
    else
        UNDEFINED;

```

MSR SPMACCESSR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x8E8] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPMACCESSR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SPMACCESSR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMACCESSR_EL2, System Performance Monitors Access Register (EL2)

The SPMACCESSR_EL2 characteristics are:

Purpose

Controls access to System PMUs from EL1 and EL0.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMACCESSR_EL2 are UNDEFINED.

Attributes

SPMACCESSR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16																
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

P<m>, bits [2m+1:2m], for m = 31 to 0

System PMU <m> access. Controls access to System PMU <m>.

P<m>	Meaning
0b00	MRS read and MSR write System register accesses to System PMU <m> at EL1 and EL0 are trapped to EL2, unless the instruction generates a higher priority exception.
0b01	MSR write System register accesses to System PMU <m> at EL1 and EL0 are trapped to EL2, unless the instruction generates a higher priority exception.
0b11	This control does not cause any instructions to be trapped.

All other values are reserved.

The registers trapped by this control are:

AArch64: [SPMCFGR_EL1](#), [SPMCGCR<n>_EL1](#), [SPMCNTENCLR_EL0](#), [SPMCNTENSET_EL0](#), [SPMCR_EL0](#), [SPMDEVAFF_EL1](#), [SPMDEVARCH_EL1](#), [SPMEVCNTR<n>_EL0](#), [SPMEVFILT2R<n>_EL0](#), [SPMEVFILTR<n>_EL0](#), [SPMEVTYPER<n>_EL0](#), [SPMIIDR_EL1](#), [SPMINTENCLR_EL1](#), [SPMINTENSET_EL1](#), [SPMOVSCCLR_EL0](#), [SPMOVSSSET_EL0](#), and [SPMSCR_EL1](#).

This field is ignored by the PE when EL2 is not implemented or disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMACCESSR_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name SPMACCESSR_EL2 or SPMACCESSR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMACCESSR_EL2

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMACCESSR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMACCESSR_EL2;

```

MSR SPMACCESSR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SPMACCESSR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMACCESSR_EL2 = X[t, 64];

```

MRS <Xt>, SPMACCESSR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E8];
    else
        X[t, 64] = SPMACCESSR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SPMACCESSR_EL2;
    else
        X[t, 64] = SPMACCESSR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMACCESSR_EL1;

```

MSR SPMACCESSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E8] = X[t, 64];
    else
        SPMACCESSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPMACCESSR_EL2 = X[t, 64];
    else
        SPMACCESSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMACCESSR_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMACCESSR_EL3, System Performance Monitors Access Register (EL3)

The SPMACCESSR_EL3 characteristics are:

Purpose

Controls access to System PMUs from EL2, EL1 and EL0.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMACCESSR_EL3 are UNDEFINED.

Attributes

SPMACCESSR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

P<m>, bits [2m+1:2m], for m = 31 to 0

System PMU <m> access. Controls access to System PMU <m>.

P<m>	Meaning
0b00	MRS read and MSR write System register accesses to System PMU <m> at EL2, EL1, and EL0 are trapped to EL3, unless the instruction generates a higher priority exception.
0b01	MSR write System register accesses to System PMU <m> at EL2, EL1, and EL0 are trapped to EL3, unless the instruction generates a higher priority exception.
0b11	This control does not cause any instructions to be trapped.

All other values are reserved.

The registers trapped by this control are:

AArch64: [SPMCFGR_EL1](#), [SPMCGCR<n>_EL1](#), [SPMCNTENCLR_EL0](#), [SPMCNTENSET_EL0](#), [SPMCR_EL0](#), [SPMDEVAFF_EL1](#), [SPMDEVARCH_EL1](#), [SPMEVCNTR<n>_EL0](#), [SPMEVFILT2R<n>_EL0](#), [SPMEVFILTR<n>_EL0](#), [SPMEVTYPER<n>_EL0](#), [SPMIIDR_EL1](#), [SPMINTENCLR_EL1](#), [SPMINTENSET_EL1](#), [SPMOVSCCLR_EL0](#), [SPMOVSSSET_EL0](#), and [SPMSCR_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMACCESSR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMACCESSR_EL3

op0	op1	CRn	CRm	op2
0b10	0b110	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMACCESSR_EL3;

```

MSR SPMACCESSR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b110	0b1001	0b1101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SPMACCESSR_EL3 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMCFGR_EL1, System Performance Monitors Configuration Register

The SPMCFGR_EL1 characteristics are:

Purpose

Describes the System Performance Monitor.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMCFGR_EL1 are UNDEFINED.

Attributes

SPMCFGR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32																																
RES0																																																															
NCG				RES0				HDBG				TRO				SS				FZO				MSI				RAO				RES0				NA				EX				RAZ				SIZE								N							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																

Bits [63:32]

Reserved, RES0.

NCG, bits [31:28]

Counter Groups.

Defines the number of counter groups implemented, minus one.

If this field is zero, then one counter group is implemented and [SPMCGCR<n>_EL1](#) read-as-zero.

Otherwise, for each counter group <m>, SPMCGCR<m DIV 8>_EL1.N<m MOD 8> defines the number of counters in the group.

Locating the first counter in each group depends on the number of implemented groups. Each counter group starts with counter:

- SPMEVTYPEPER<m×32>_EL0, meaning there are at most 32 counters per group, if there are 2 counter groups.
- SPMEVTYPEPER<m×16>_EL0, meaning there are at most 16 counters per group, if there are 3 or 4 counter groups.
- SPMEVTYPEPER<m×8>_EL0, meaning there are at most 8 counters per group, if there are between 5 and 8 counter groups.
- SPMEVTYPEPER<m×4>_EL0, meaning there are at most 4 counters per group, if there are more than 8 counter groups.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [27:25]

Reserved, RES0.

HDBG, bit [24]

Halt-on-debug supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

TRO, bit [23]

Trace output supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SS, bit [22]

Snapshot supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

FZO, bit [21]

Freeze-on-overflow supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

MSI, bit [20]

Message-signaled interrupts supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bit [19]

Reserved, RAO.

Bit [18]

Reserved, RES0.

NA, bit [17]

No write access when running. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

EX, bit [16]

Export supported. For more information on this field, see 'CoreSight PMU Architecture'.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [15:14]

Reserved, RAZ.

SIZE, bits [13:8]

Counter size. The size of the largest implemented counter.

SIZE	Meaning
0b000111	8-bit counters.
0b001001	10-bit counters.
0b001011	12-bit counters.
0b001111	16-bit counters.
0b010011	20-bit counters.
0b010111	24-bit counters.
0b011111	32-bit counters.
0b100011	36-bit counters.
0b100111	40-bit counters.
0b101011	44-bit counters.
0b101111	48-bit counters.
0b110011	52-bit counters.
0b110111	56-bit counters.
0b111111	64-bit counters.

All other values are reserved.

Not all counters must be this size. For example, an implementation might include a mix of 32-bit and 64-bit counters.

N, bits [7:0]

Number of event counters.

Accessing SPMCFGR_EL1

To access SPMCFGR_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

SPMCFGR_EL1 reads-as-zero if the System PMU selected by [SPMSELR_EL0](#).SYSPMUSEL is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMCFGR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMCFGR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMCFGR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMCFGR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

no old file

htmldiff from-

(new)

SPMCGCR<n>_EL1, Counter Group Configuration Register <n>, n = 0 - 1

The SPMCGCR<n>_EL1 characteristics are:

Purpose

Describes the performance monitor.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMCGCR<n>_EL1 are UNDEFINED.

Attributes

SPMCGCR<n>_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
N7								N6								N5								N4							
N3								N2								N1								N0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

N<m>, bits [8m+7:8m], for m = 7 to 0

Number of counters in group $n \times 8 + m$.

The maximum size of each counter group depends on the number of implemented groups and the largest implemented counter size. For more information, see [SPMCFGR_EL1.NCG](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMCGCR<n>_EL1

To access SPMCGCR<n>_EL1 for System PMU <s>, set [SPMSELR_EL0.SYSPMUSEL](#) to s.

SPMCGCR<n>_EL1 reads-as-zero if any of the following are true:

- The System PMU selected by [SPMSELR_EL0.SYSPMUSEL](#) is not implemented.
- [SPMCFGR_EL1.NCG](#) is zero.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMCGCR<n>_EL1 ; Where n = 0-1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b00:n[0]

```
integer n = UInt(op2<0>);  
  
if PSTATE.EL == EL0 then  
    UNDEFINED;  
elsif PSTATE.EL == EL1 then  
    X[t, 64] = SPMCGCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];  
elsif PSTATE.EL == EL2 then  
    X[t, 64] = SPMCGCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];  
elsif PSTATE.EL == EL3 then  
    X[t, 64] = SPMCGCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

SPMCNTENCLR_EL0, System Performance Monitors Count Enable Clear Register

The SPMCNTENCLR_EL0 characteristics are:

Purpose

Disables implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMCNTENCLR_EL0 are UNDEFINED.

Attributes

SPMCNTENCLR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34	P33	P32	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<m>, bit [m], for m = 63 to 0

Event counter <m> disable.

P<m>	Meaning
0b0	Event counter <m> is disabled.
0b1	Event counter <m> is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1C**.

Accessing SPMCNTENCLR_EL0

To access SPMCNTENCLR_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMCNTENCLR_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b010


```

if PSTATE.EL == EL0 then
    X[t, 64] = SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMCNTENCLR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b010

```

if PSTATE.EL == EL0 then
    SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMCNTENCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file htmldiff from- (new)

SPMCNTENSET_EL0, System Performance Monitors Count Enable Set Register

The SPMCNTENSET_EL0 characteristics are:

Purpose

Enables implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMCNTENSET_EL0 are UNDEFINED.

Attributes

SPMCNTENSET_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34	P33	P32	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<m>, bit [m], for m = 63 to 0

Event counter <m> enable.

P<m>	Meaning
0b0	Event counter <m> is disabled.
0b1	Event counter <m> is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1S**.

Accessing SPMCNTENSET_EL0

To access SPMCNTENSET_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMCNTENSET_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    X[t, 64] = SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMCNTENSET_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMCNTENSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMCR_EL0, System Performance Monitor Control Register

The SPMCR_EL0 characteristics are:

Purpose

Main control register for the System Performance Monitors.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMCR_EL0 are UNDEFINED.

Attributes

SPMCR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
																RES0																	
RES0												TRO				HDBG				FZONA				RES0				EX		RES0		P	E
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:12]

Reserved, RES0.

TRO, bit [11]

When SPMCFGR_EL1.TRO == 1:

Trace enable. For more information on this field, see 'CoreSight PMU Architecture'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When PSTATE.EL == EL0, access to this field is **RO**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

HDBG, bit [10]

When SPMCFGR_EL1.HDBG == 1:

Halt-on-debug. For more information on this field, see 'CoreSight PMU Architecture'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When PSTATE.EL == EL0, access to this field is **RO**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

FZO, bit [9]

When SPMCFGR_EL1.FZO == 1:

Freeze-on-overflow. For more information on this field, see 'CoreSight PMU Architecture'.

Note that, if implemented by a System PMU, then freeze-on-overflow affects only the counters of that System PMU, not other System PMUs nor the PE PMU.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NA, bit [8]

When SPMCFGR_EL1.NA == 1:

Not accessible. For more information on this field, see 'CoreSight PMU Architecture'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [7:5]

Reserved, RES0.

EX, bit [4]

When SPMCFGR_EL1.EX == 1:

Export enable. For more information on this field, see 'CoreSight PMU Architecture'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [3:2]

Reserved, RES0.

P, bit [1]

Event counter reset.

P	Meaning
0b0	Write is ignored.
0b1	Reset all event counters to zero. If the cycle counter is implemented, the cycle counter is not reset.

Note

Resetting the event counters does not affect any overflow flags.

Access to this field is **WO/RAZ**.

E, bit [0]

Count enable. This bit controls the System Performance Monitor.

E	Meaning
0b0	Monitor is disabled.
0b1	Monitor is enabled.

Performance monitor overflow IRQs are only signaled when this bit is set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SPMCR_EL0

To access SPMCR_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMCR_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b000

```

if PSTATE.EL == EL0 then
    X[t, 64] = SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b000

```

if PSTATE.EL == EL0 then
    SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMCR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMDEVAFF_EL1, System Performance Monitors Device Affinity Register

The SPMDEVAFF_EL1 characteristics are:

Purpose

For additional information, see the CoreSight Architecture Specification.

For a System PMU that has affinity with a single PE or a group of PEs, SPMDEVAFF_EL1 is a copy of [MPIDR_EL1](#) or part of [MPIDR_EL1](#):

- If the System PMU has affinity with a single PE, the affinity level is 0, then SPMDEVAFF_EL1 reads the same value as [MPIDR_EL1](#), and SPMDEVAFF_EL1.FOV reads-as-one to indicate affinity level 0.
- If the System PMU has affinity with a group of PEs, the affinity level is 1, 2, or 3, then parts of SPMDEVAFF_EL1 reads the same value as parts of [MPIDR_EL1](#), and the rest of SPMDEVAFF_EL1 indicates the level.

For example, if the group of PEs is a subset of the PEs at affinity level 1 then all of the following are true:

- All the PEs in the group have the same values in [MPIDR_EL1](#).{Aff3,Aff2}, and these values are equal to SPMDEVAFF_EL1.{Aff3,Aff2}.
- SPMDEVAFF_EL1.Aff1 is nonzero and not 0x80, and SPMDEVAFF_EL1.{Aff0,FOV} read-as-zero, to indicate at least affinity level 1. The subset of PEs at level 1 that the System PMU has affinity with is indicated by the least-significant set bit in SPMDEVAFF_EL1.Aff1. In this example, if SPMDEVAFF_EL1.Aff1[2:0] is 0b100, then the System PMU has affinity with the up-to 8 PEs that have [MPIDR_EL1](#).Aff1[7:3] == SPMDEVAFF_EL1.Aff1[7:3].

Depending on the IMPLEMENTATION DEFINED nature of the system, it might be possible that SPMDEVAFF_EL1 is read before system firmware has configured the System PMU and/or the PE or group of PEs that the System PMU has affinity with. When this is the case, SPMDEVAFF_EL1 reads as zero.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMDEVAFF_EL1 are UNDEFINED.

Attributes

SPMDEVAFF_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																								Aff3							
FOV		U	RES0						MT	Aff2								Aff1								Aff0					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

PE affinity level 3. The [MPIDR_EL1](#).Aff3 field, viewed from the highest Exception level of the associated PE or PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

F0V, bit [31]

Indicates that the SPMDEVAFF_EL1.Aff0 field is valid.

F0V	Meaning
0b0	SPMDEVAFF_EL1.Aff0 is not valid, and the PE affinity is above level 0 or a subset of level 0.
0b1	SPMDEVAFF_EL1.Aff0 is valid, and the PE affinity is at level 0.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

U, bit [30]

When SPMDEVAFF_EL1.F0V == 1:

Uniprocessor. The [MPIDR_EL1](#).U field, viewed from the highest Exception level of the associated PE.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, UNKNOWN.

Bits [29:25]

Reserved, RES0.

MT, bit [24]

When SPMDEVAFF_EL1.F0V == 1:

Multithreaded. The [MPIDR_EL1](#).MT field, viewed from the highest Exception level of the associated PE.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, UNKNOWN.

Aff2, bits [23:16]

When affine with a PE or PEs at affinity level 2 or below:

PE affinity level 2. The [MPIDR_EL1](#).Aff2 field, viewed from the highest Exception level of the associated PE or PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

When affine with a sub-set of PEs at affinity level 2:

PE affinity level 2. Defines part of the [MPIDR_EL1](#).Aff2 field, viewed from the highest Exception level of the associated PEs.

Aff2	Meaning
0bxxxxxxx1	SPMDEVAFF_EL1.Aff2[7:1] is the value of MPIDR_EL1 .Aff2[7:1], viewed from the highest Exception level of the associated PEs.
0bxxxxxx10	SPMDEVAFF_EL1.Aff2[7:2] is the value of MPIDR_EL1 .Aff2[7:2], viewed from the highest Exception level of the associated PEs.
0bxxxxx100	SPMDEVAFF_EL1.Aff2[7:3] is the value of MPIDR_EL1 .Aff2[7:3], viewed from the highest Exception level of the associated PEs.
0bxxxx1000	SPMDEVAFF_EL1.Aff2[7:4] is the value of MPIDR_EL1 .Aff2[7:4], viewed from the highest Exception level of the associated PEs.
0bxxx10000	SPMDEVAFF_EL1.Aff2[7:5] is the value of MPIDR_EL1 .Aff2[7:5], viewed from the highest Exception level of the associated PEs.
0bxx100000	SPMDEVAFF_EL1.Aff2[7:6] is the value of MPIDR_EL1 .Aff2[7:6], viewed from the highest Exception level of the associated PEs.
0bx1000000	SPMDEVAFF_EL1.Aff2[7] is the value of MPIDR_EL1 .Aff2[7], viewed from the highest Exception level of the associated PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

PE affinity level NOT DEFINED. Indicates whether the PE affinity is at level 3.

Aff2	Meaning
0x80	PE affinity is at level 3.

All other values are reserved.

Access to this field is **RO**.

Aff1, bits [15:8]**When affine with a PE or PEs at affinity level 1 or below:**

PE affinity level 1. The [MPIDR_EL1](#).Aff1 field, viewed from the highest Exception level of the associated PE or PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

When affine with a sub-set of PEs at affinity level 1:

PE affinity level 1. Defines part of the [MPIDR_EL1](#).Aff1 field, viewed from the highest Exception level of the associated PEs.

Aff1	Meaning
0bxxxxxxx1	SPMDEVAFF_EL1.Aff1[7:1] is the value of MPIDR_EL1 .Aff1[7:1], viewed from the highest Exception level of the associated PEs.
0bxxxxxx10	SPMDEVAFF_EL1.Aff1[7:2] is the value of MPIDR_EL1 .Aff1[7:2], viewed from the highest Exception level of the associated PEs.
0bxxxxx100	SPMDEVAFF_EL1.Aff1[7:3] is the value of MPIDR_EL1 .Aff1[7:3], viewed from the highest Exception level of the associated PEs.
0bxxxx1000	SPMDEVAFF_EL1.Aff1[7:4] is the value of MPIDR_EL1 .Aff1[7:4], viewed from the highest Exception level of the associated PEs.
0bxxx10000	SPMDEVAFF_EL1.Aff1[7:5] is the value of MPIDR_EL1 .Aff1[7:5], viewed from the highest Exception level of the associated PEs.
0bxx100000	SPMDEVAFF_EL1.Aff1[7:6] is the value of MPIDR_EL1 .Aff1[7:6], viewed from the highest Exception level of the associated PEs.
0bx1000000	SPMDEVAFF_EL1.Aff1[7] is the value of MPIDR_EL1 .Aff1[7], viewed from the highest Exception level of the associated PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

PE affinity level 1. Indicates whether the PE affinity is at level 2.

Aff1	Meaning
0x00	PE affinity is above level 2 or a subset of level 2.
0x80	PE affinity is at level 2.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Aff0, bits [7:0]

When affine with a PE at affinity level 0:

PE affinity level 0. The [MPIDR_EL1](#).Aff0 field, viewed from the highest Exception level of the associated PE.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

When affine with a sub-set of PEs at affinity level 0:

PE affinity level 0. Defines part of the [MPIDR_EL1](#).Aff0 field, viewed from the highest Exception level of the associated PEs.

Aff0	Meaning
0bxxxxxxx1	SPMDEVAFF_EL1.Aff0[7:1] is the value of MPIDR_EL1 .Aff0[7:1], viewed from the highest Exception level of the associated PEs.
0bxxxxxx10	SPMDEVAFF_EL1.Aff0[7:2] is the value of MPIDR_EL1 .Aff0[7:2], viewed from the highest Exception level of the associated PEs.
0bxxxxx100	SPMDEVAFF_EL1.Aff0[7:3] is the value of MPIDR_EL1 .Aff0[7:3], viewed from the highest Exception level of the associated PEs.
0bxxxx1000	SPMDEVAFF_EL1.Aff0[7:4] is the value of MPIDR_EL1 .Aff0[7:4], viewed from the highest Exception level of the associated PEs.
0bxxx10000	SPMDEVAFF_EL1.Aff0[7:5] is the value of MPIDR_EL1 .Aff0[7:5], viewed from the highest Exception level of the associated PEs.
0bxx100000	SPMDEVAFF_EL1.Aff0[7:6] is the value of MPIDR_EL1 .Aff0[7:6], viewed from the highest Exception level of the associated PEs.
0bx1000000	SPMDEVAFF_EL1.Aff0[7] is the value of MPIDR_EL1 .Aff0[7], viewed from the highest Exception level of the associated PEs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

PE affinity level 0. Indicates whether the PE affinity is at level 1.

Aff0	Meaning
0x00	PE affinity is above level 1 or a subset of level 1.
0x80	PE affinity is at level 1.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing SPMDEVAFF_EL1

Reads of SPMDEVAFF_EL1 are not affected by the value of [VMPIDR_EL2](#) at any Exception level.

If the System PMU has affinity only with this PE, then it is IMPLEMENTATION DEFINED whether SPMDEVAFF_EL1 reads-as-zero or reads the same value as [MPIDR_EL1](#).

To access SPMDEVAFF_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

SPMDEVAFF_EL1 reads-as-zero if the System PMU selected by [SPMSELR_EL0](#).SYSPMUSEL is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMDEVAFF_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMDEVAFF_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMDEVAFF_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMDEVAFF_EL1;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

no old file

htmldiff from-

(new)

SPMDEVARCH_EL1, System Performance Monitors Device Architecture Register

The SPMDEVARCH_EL1 characteristics are:

Purpose

Provides discovery information for the component.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMDEVARCH_EL1 are UNDEFINED.

Attributes

SPMDEVARCH_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

ARCHITECT, bits [31:21]

Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PRESENT, bit [20]

DEVARCH present. Defines that SPMDEVARCH_EL1 register is present.

PRESENT	Meaning
0b0	Device Architecture information not present.
0b1	Device Architecture information present.

This bit reads as 1.

REVISION, bits [19:16]

Revision. Defines the architecture revision of the component.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component.

SPMDEVARCH_EL1.ARCHVER and SPMDEVARCH_EL1.ARCHPART are also defined as a single field, SPMDEVARCH_EL1.ARCHID, so that SPMDEVARCH_EL1.ARCHVER is SPMDEVARCH_EL1.ARCHID[15:12].

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

SPMDEVARCH_EL1.ARCHVER and SPMDEVARCH_EL1.ARCHPART are also defined as a single field, SPMDEVARCH_EL1.ARCHID, so that SPMDEVARCH_EL1.ARCHPART is SPMDEVARCH_EL1.ARCHID[11:0].

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing SPMDEVARCH_EL1

To access SPMDEVARCH_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

SPMDEVARCH_EL1 reads-as-zero if the System PMU selected by [SPMSELR_EL0](#).SYSPMUSEL is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMDEVARCH_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMDEVARCH_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMDEVARCH_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMDEVARCH_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMEVCNTR<n>_EL0, System Performance Monitors Event Count Register, n = 0 - 63

The SPMEVCNTR<n>_EL0 characteristics are:

Purpose

Holds event counter n, which counts events, where n is 0 to 15.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMEVCNTR<n>_EL0 are UNDEFINED.

Attributes

SPMEVCNTR<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
CNTR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTR																															

CNTR, bits [63:0]

Event counter n.

The number of implemented bits for SPMEVCNTR<n>_EL0 is IMPLEMENTATION DEFINED. Unimplemented bits are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMEVCNTR<n>_EL0

To access SPMEVCNTR<n>_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s and [SPMSELR_EL0](#).BANK to n[5:4].

SPMEVCNTR<n>_EL0 reads-as-zero and ignores writes if event counter <n> is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMEVCNTR<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b000:m[3]	m[2:0]


```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    X[t, 64] = SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
```

MSR SPMEVCNTR<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b000:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMEVCNTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMEVFILT2R<n>_EL0, Additional System Performance Monitors Event Filter Control Register 2, n = 0 - 63

The SPMEVFILT2R<n>_EL0 characteristics are:

Purpose

For System Performance Monitors requiring additional event selection controls.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMEVFILT2R<n>_EL0 are UNDEFINED.

Attributes

SPMEVFILT2R<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing SPMEVFILT2R<n>_EL0

To access SPMEVFILT2R<n>_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s and [SPMSELR_EL0](#).BANK to n[5:4].

SPMEVFILT2R<n>_EL0 reads-as-zero and ignores writes if event counter <n> is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMEVFILT2R<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b011:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    X[t, 64] = SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
```

MSR SPMEVFILT2R<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b011:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMEVFILT2R_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMEVFILTR<n>_EL0, Additional System Performance Monitors Event Filter Control Register, n = 0 - 63

The SPMEVFILTR<n>_EL0 characteristics are:

Purpose

For System Performance Monitors requiring additional event selection controls.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMEVFILTR<n>_EL0 are UNDEFINED.

Attributes

SPMEVFILTR<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing SPMEVFILTR<n>_EL0

To access SPMEVFILTR<n>_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s and [SPMSELR_EL0](#).BANK to n[5:4].

SPMEVFILTR<n>_EL0 reads-as-zero and ignores writes if event counter <n> is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMEVFILTR<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b010:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    X[t, 64] = SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
```

MSR SPMEVFILTR<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b010:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMEVFILTR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMEVTYPEPER<n>_EL0, System Performance Monitors Event Type Register, n = 0 - 63

The SPMEVTYPEPER<n>_EL0 characteristics are:

Purpose

Configures which event increments the event counter [SPMEVCNTR<n>_EL0](#), or to read the current configuration.

The contents of this register are IMPLEMENTATION DEFINED. An Event Type Select Register typically contains:

- A field defining the event that the counter is responsive to, in the least-significant bits.
- Controls for per-counter filtering, such as by mode or state.
- Additional controls, such as for a per-counter state machine.

Additional controls might also be provided in the [SPMEVFILTR<n>_EL0](#) and [SPMEVFILT2R<n>_EL0](#) registers.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMEVTYPEPER<n>_EL0 are UNDEFINED.

Attributes

SPMEVTYPEPER<n>_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

Accessing SPMEVTYPEPER<n>_EL0

To access SPMEVTYPEPER<n>_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s and [SPMSELR_EL0](#).BANK to n[5:4].

SPMEVTYPEPER<n>_EL0 reads-as-zero and ignores writes if event counter <n> is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMEVTYPEPER<m>_EL0 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b001:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    X[t, 64] = SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m];
```

MSR SPMEVTYPEPER<m>_EL0, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b011	0b1110	0b001:m[3]	m[2:0]

```
integer m = UInt(CRm<0>:op2<2:0>);

if PSTATE.EL == EL0 then
    SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMEVTYPEPER_EL0[UInt(SPMSELR_EL0.SYSPMUSEL), m] = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file	htmldiff from-	(new)
-------------	----------------	-------

no old file

htmldiff from-

(new)

SPMIIDR_EL1, Implementation Identification Register

The SPMIIDR_EL1 characteristics are:

Purpose

Provides discovery information about the component.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMIIDR_EL1 are UNDEFINED.

Attributes

SPMIIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
RES0																																			
ProductID																Variant				Revision				Implementer[10:7]				RES0		Implementer[6:0]					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:32]

Reserved, RES0.

ProductID, bits [31:20]

Part number, bits [11:0]. The part number is selected by the designer of the component.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Variant, bits [19:16]

Component major revision.

Defines either a variant of the component defined by SPMIIDR_EL1.ProductID, or the major revision of the component.

When defining a major revision, SPMIIDR_EL1.Variant and SPMIIDR_EL1.Revision together form the revision number of the component, with SPMIIDR_EL1.Variant being the most significant part and SPMIIDR_EL1.Revision the least significant part. When a component is changed, SPMIIDR_EL1.Variant or SPMIIDR_EL1.Revision is increased to ensure that software can differentiate the different revisions of the component. If SPMIIDR_EL1.Variant is increased then SPMIIDR_EL1.Revision should be set to 0b0000.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Component minor revision.

When a component is changed:

- If SPMIIDR_EL1.Variant and SPMIIDR_EL1.Revision together form the revision number of the component then:
 - SPMIIDR_EL1.Variant or SPMIIDR_EL1.Revision is increased to ensure that software can differentiate the different revisions of the component.
 - If Variant is increased then Revision should be set to 0b0000.
- Otherwise, SPMIIDR_EL1.Revision is increased to ensure that software can differentiate the different revisions of the component.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:8, 6:0]

JEDEC-assigned JEP106 identification code of the designer of the component.

SPMIIDR_EL1[11:8] is the JEP106 bank identifier minus 1 and SPMIIDR_EL1[6:0] is the JEP106 identification code for the designer of the component. The code identifies the designer of the component, which might not be the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For example, for a component designed by Arm Limited, the JEP106 bank is 5, and the JEP106 identification code is 0x3B, meaning SPMIIDR_EL1[11:0] has the value 0x43B.

Zero is not a valid JEP106 identification code, meaning a value of zero for SPMIIDR_EL1 indicates this register is not implemented.

This field has an IMPLEMENTATION DEFINED value.

The Implementer field is split as follows:

- Implementer[10:7] is SPMIIDR_EL1[11:8].
- Implementer[6:0] is SPMIIDR_EL1[6:0].

Access to this field is **RO**.

Bit [7]

Reserved, RES0.

Accessing SPMIIDR_EL1

To access SPMIIDR_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

SPMIIDR_EL1 reads-as-zero if the System PMU selected by [SPMSELR_EL0](#).SYSPMUSEL is not implemented.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMIIDR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1101	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMIIDR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMIIDR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMIIDR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

SPMINTENCLR_EL1, System Performance Monitors Interrupt Enable Clear Register

The SPMINTENCLR_EL1 characteristics are:

Purpose

Disables the generation of interrupt requests on overflows from implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMINTENCLR_EL1 are UNDEFINED.

Attributes

SPMINTENCLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

P<m>, bit [m], for m = 63 to 0

Event counter <m> overflow interrupt request disable.

P<m>	Meaning
0b0	Event counter <m> interrupt request is disabled.
0b1	Event counter <m> interrupt request is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1C**.

Accessing SPMINTENCLR_EL1

To access SPMINTENCLR_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMINTENCLR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1110	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
```

MSR SPMINTENCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1110	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMINTENCLR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMINTENSET_EL1, System Performance Monitors Interrupt Enable Set Register

The SPMINTENSET_EL1 characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMINTENSET_EL1 are UNDEFINED.

Attributes

SPMINTENSET_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34	P33	P32	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<m>, bit [m], for m = 63 to 0

Event counter <m> overflow interrupt request enable.

P<m>	Meaning
0b0	Event counter <m> interrupt request is disabled.
0b1	Event counter <m> interrupt request is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1S**.

Accessing SPMINTENSET_EL1

To access SPMINTENSET_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMINTENSET_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMINTENSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b1001	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMINTENSET_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMOVSLR_EL0, System Performance Monitors Overflow Flag Status Clear Register

The SPMOVSLR_EL0 characteristics are:

Purpose

Clears the state of the overflow bit for the implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMOVSLR_EL0 are UNDEFINED.

Attributes

SPMOVSLR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34	P33	P32	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<m>, bit [m], for m = 63 to 0

Event counter <m> unsigned overflow bit clear.

P<m>	Meaning
0b0	Event counter <m> has not overflowed.
0b1	Event counter <m> has overflowed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1C**.

Accessing SPMOVSLR_EL0

To access SPMOVSLR_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMOVSLR_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b011

```

if PSTATE.EL == EL0 then
    X[t, 64] = SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMOVSCLR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b011

```

if PSTATE.EL == EL0 then
    SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMOVSCLR_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMOVSSET_EL0, System Performance Monitors Overflow Flag Status Set Register

The SPMOVSSET_EL0 characteristics are:

Purpose

Sets the state of the overflow bit for the implemented event counters.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMOVSSET_EL0 are UNDEFINED.

Attributes

SPMOVSSET_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P63	P62	P61	P60	P59	P58	P57	P56	P55	P54	P53	P52	P51	P50	P49	P48	P47	P46	P45	P44	P43	P42	P41	P40	P39	P38	P37	P36	P35	P34	P33	P32	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<m>, bit [m], for m = 63 to 0

Event counter <m> unsigned overflow bit set.

P<m>	Meaning
0b0	Event counter <m> has not overflowed.
0b1	Event counter <m> has overflowed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When event counter <m> is not implemented, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **W1S**.

Accessing SPMOVSSET_EL0

To access SPMOVSSET_EL0 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMOVSSET_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    X[t, 64] = SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMOVSSET_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMOVSSET_EL0[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

SPMROOTCR_EL3, System Performance Monitors Root and Realm Control Register

The SPMROOTCR_EL3 characteristics are:

Purpose

Controls observability of Root and Realm events by the System Performance Monitor.

Configuration

This register is present only when FEAT_RME is implemented and FEAT_SPMU is implemented. Otherwise, direct accesses to SPMROOTCR_EL3 are UNDEFINED.

Attributes

SPMROOTCR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPL	RES0																											NAO	RES0	RL	ORTO
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:32]

IMPLEMENTATION DEFINED observation controls. Additional IMPLEMENTATION DEFINED bits to control certain types of filter or events.

IMPL, bit [31]

Indicates SPMROOTCR_EL3 is present.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

Bits [30:4]

Reserved, RES0.

NAO, bit [3]

When the System PMU can count or monitor non-attributable events:

Non-attributable Observation. Controls whether events or monitorable characteristics not attributable with any source can be monitored.

NAO	Meaning
0b0	Events not attributable with any event source are not counted.
0b1	Counting non-attributable events is not prevented by this bit.

When both SPMROOTCR_EL3 and [SPMSCR_EL1](#) are implemented, non-attributable events are counted only if both SPMROOTCR_EL3.NAO is 1 and [SPMSCR_EL1](#).{NAO, SO} is nonzero.

SPMROOTCR_EL3.NAO has the opposite reset polarity to [SPMSCR_EL1](#).NAO.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Otherwise:

Reserved, RES0.

Bit [2]

Reserved, RES0.

RLO, bit [1]

Realm Observation. Controls whether events or monitorable characteristics attributable to a Realm event source can be monitored.

RLO	Meaning
0b0	Events attributable to a Realm event source are not counted.
0b1	Events attributable to a Realm event source are counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

RTO, bit [0]

Root Observation. Controls whether events or monitorable characteristics attributable to a Root event source can be monitored.

RTO	Meaning
0b0	Events attributable to a Root event source are not counted.
0b1	Events attributable to a Root event source are counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SPMROOTCR_EL3

To access SPMROOTCR_EL3 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMROOTCR_EL3

op0	op1	CRn	CRm	op2
0b10	0b110	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMROOTCR_EL3[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMROOTCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b110	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SPMROOTCR_EL3[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMSCR_EL1, System Performance Monitors Secure Control Register

The SPMSCR_EL1 characteristics are:

Purpose

Controls observability of Secure events by the System Performance Monitor, and optionally controls Secure attributes for message signaled interrupts and Non-secure access to the performance monitor registers.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMSCR_EL1 are UNDEFINED.

Attributes

SPMSCR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMPLEMENTATION DEFINED																															
IMPL	RES0																										NAO	RES0	SC		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IMPLEMENTATION DEFINED, bits [63:32]

IMPLEMENTATION DEFINED observation controls. Additional IMPLEMENTATION DEFINED bits to control certain types of filter or events.

IMPL, bit [31]

Indicates SPMSCR_EL1 is present.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

Bits [30:5]

Reserved, RES0.

NAO, bit [4]

When the System PMU can count or monitor non-attributable events:

Non-attributable Observation. Controls whether events or monitorable characteristics not attributable with any source can be monitored.

NAO	Meaning
0b0	Events not attributable with any event source are not counted, unless overridden by SPMSCR_EL1.SO.
0b1	Counting non-attributable events is not prevented by this bit.

When both [SPMROOTCR_EL3](#) and SPMSCR_EL1 are implemented, non-attributable events are counted only if both [SPMROOTCR_EL3](#).NAO is 1 and SPMSCR_EL1.{NAO, SO} is nonzero.

SPMSCR_EL1.NAO has the opposite reset polarity to [SPMROOTCR_EL3](#).NAO.

This bit is optional if Root and Realm states are not implemented. When this bit is not implemented, the PMU behaves as if SPMSCR_EL1.NAO is 0, and whether events or monitorable characteristics not attributable with any source can be monitored is controlled by SPMSCR_EL1.SO.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [3:1]

Reserved, RES0.

SO, bit [0]

Secure Observation. Controls whether events or monitorable characteristics attributable to a Secure event source can be monitored.

SO	Meaning
0b0	Events attributable to a Secure event source are not counted.
0b1	Events attributable to a Secure event source are counted.

Also controls whether events or monitorable characteristics not attributable with any source can be monitored. See SPMSCR_EL1.NAO.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SPMSCR_EL1

To access SPMSCR_EL1 for System PMU <s>, set [SPMSELR_EL0](#).SYSPMUSEL to s.

SPMSCR_EL1 is UNDEFINED if accessed in Non-secure or Realm state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMSCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b111	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)];

```

MSR SPMSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b111	0b1001	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMSCR_EL1[UInt(SPMSELR_EL0.SYSPMUSEL)] = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

SPMSELR_EL0, System Performance Monitors Select Register

The SPMSELR_EL0 characteristics are:

Purpose

Selects the System PMU and event counter registers to access.

Configuration

This register is present only when FEAT_SPMU is implemented. Otherwise, direct accesses to SPMSELR_EL0 are UNDEFINED.

Attributes

SPMSELR_EL0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																RES0								SYSPMUSEL				RES0		BANK	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:10]

Reserved, RES0.

SYSPMUSEL, bits [9:4]

System PMU Select. Selects a System PMU <s> to access.

Values 0x20 to 0x3F are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [3:2]

Reserved, RES0.

BANK, bits [1:0]

System PMU bank access control. Selects a bank of 16 System PMU event counters and related controls to access.

BANK	Meaning
0b00	Select event counters 0 to 15.
0b01	Select event counters 16 to 31.
0b10	Select event counters 32 to 47.
0b11	Select event counters 48 to 63.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPMSELR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPMSELR_EL0

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b101

```
if PSTATE.EL == EL0 then
    X[t, 64] = SPMSELR_EL0;
elsif PSTATE.EL == EL1 then
    X[t, 64] = SPMSELR_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPMSELR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPMSELR_EL0;
```

MSR SPMSELR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b1001	0b1100	0b101

```
if PSTATE.EL == EL0 then
    SPMSELR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    SPMSELR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    SPMSELR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPMSELR_EL0 = X[t, 64];
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SPSR_EL1, Saved Program Status Register (EL1)

The SPSR_EL1 characteristics are:

Purpose

Holds the saved process state when an exception is taken to EL1.

Configuration

AArch64 System register SPSR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [SPSR_svc\[31:0\]](#).

Attributes

SPSR_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported and exception taken from AArch32 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
N	Z	C	V	Q	IT[1:0]	DIT	SSBS	PAN	SS	IL	GE	IT[7:2]	E	A	I	F	T	M[4]	M[3:0]												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

An exception return from EL1 using AArch64 makes SPSR_EL1 become UNKNOWN.

Bits [63:32]

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL1, and copied to PSTATE.N on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL1, and copied to PSTATE.Z on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL1, and copied to PSTATE.C on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL1, and copied to PSTATE.V on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Q, bit [27]

Overflow or saturation flag. Set to the value of PSTATE.Q on taking an exception to EL1, and copied to PSTATE.Q on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IT, bits [15:10, 26:25]

If-Then. Set to the value of PSTATE.IT on taking an exception to EL1, and copied to PSTATE.IT on executing an exception return operation in EL1.

SPSR_EL1.IT must contain a value that is valid for the instruction being returned to.

The IT field is split as follows:

- IT[1:0] is SPSR_EL1[26:25].
- IT[7:2] is SPSR_EL1[15:10].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIT, bit [24]

When FEAT_DIT is implemented:

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL1, and copied to PSTATE.DIT on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [23]

When FEAT_SSBS is implemented:

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL1, and copied to PSTATE.SSBS on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL1, and copied to PSTATE.PAN on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL1, and conditionally copied to PSTATE.SS on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL1, and copied to PSTATE.IL on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GE, bits [19:16]

Greater than or Equal flags. Set to the value of PSTATE.GE on taking an exception to EL1, and copied to PSTATE.GE on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E, bit [9]

Endianness. Set to the value of PSTATE.E on taking an exception to EL1, and copied to PSTATE.E on executing an exception return operation in EL1.

If the implementation does not support big-endian operation, SPSR_EL1.E is RES0. If the implementation does not support little-endian operation, SPSR_EL1.E is RES1. On executing an exception return operation in EL1, if the implementation does not support big-endian operation at the Exception level being returned to, SPSR_EL1.E is RES0, and if the implementation does not support little-endian operation at the Exception level being returned to, SPSR_EL1.E is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL1, and copied to PSTATE.A on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL1, and copied to PSTATE.I on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL1, and copied to PSTATE.F on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T, bit [5]

T32 Instruction set state. Set to the value of PSTATE.T on taking an exception to EL1, and copied to PSTATE.T on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[4], bit [4]

Execution state. Set to 0b1, the value of PSTATE.nRW, on taking an exception to EL1 from AArch32 state, and copied to PSTATE.nRW on executing an exception return operation in EL1.

M[4]	Meaning
0b1	AArch32 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch32 Mode. Set to the value of PSTATE.M[3:0] on taking an exception to EL1, and copied to PSTATE.M[3:0] on executing an exception return operation in EL1.

M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0111	Abort.
0b1011	Undefined.
0b1111	System.

Other values are reserved. If SPSR_EL1.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL1 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When exception taken from AArch64 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
PENDPM																															
N	Z	C	V	RES0	TCODIT	UAO	PAN	SS	IL	RES0				ALLINT	SSBS	BTYPE	D	A	I	F	RES0	M[4]	M[3:0]								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

An exception return from EL1 using AArch64 makes SPSR_EL1 become UNKNOWN.

Bits [63:34:32]

Reserved, RES0.

PPEND, bit [33]

When FEAT_SEBEP is implemented:

PMU exception pending bit. Set to the value of PSTATE.PPEND on taking an exception to EL1, and conditionally copied to PSTATE.PPEND on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PM, bit [32]

When FEAT_EBEP is implemented:

PMU exception mask bit. Set to the value of PSTATE.PM on taking an exception to EL1, and copied to PSTATE.PM on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL1, and copied to PSTATE.N on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL1, and copied to PSTATE.Z on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL1, and copied to PSTATE.C on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL1, and copied to PSTATE.V on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

TCO, bit [25]

When FEAT_MTE is implemented:

Tag Check Override. Set to the value of PSTATE.TCO on taking an exception to EL1, and copied to PSTATE.TCO on executing an exception return operation in EL1.

When FEAT_MTE2 is not implemented, it is CONSTRAINED UNPREDICTABLE whether this field is RES0 or behaves as if **FEAT_MTE2** ~~FEAT_MTE~~ is implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DIT, bit [24]

When FEAT_DIT is implemented:

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL1, and copied to PSTATE.DIT on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UAO, bit [23]**When FEAT_UAO is implemented:**

User Access Override. Set to the value of PSTATE.UAO on taking an exception to EL1, and copied to PSTATE.UAO on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL1, and copied to PSTATE.PAN on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL1, and conditionally copied to PSTATE.SS on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL1, and copied to PSTATE.IL on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:14]

Reserved, RES0.

ALLINT, bit [13]**When FEAT_NMI is implemented:**

All IRQ or FIQ interrupts mask. Set to the value of PSTATE.ALLINT on taking an exception to EL1, and copied to PSTATE.ALLINT on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [12]**When FEAT_SSBS is implemented:**

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL1, and copied to PSTATE.SSBS on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BTYPE, bits [11:10]**When FEAT_BTI is implemented:**

Branch Type Indicator. Set to the value of PSTATE.BTYPE on taking an exception to EL1, and copied to PSTATE.BTYPE on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D, bit [9]

Debug exception mask. Set to the value of PSTATE.D on taking an exception to EL1, and copied to PSTATE.D on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL1, and copied to PSTATE.A on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL1, and copied to PSTATE.I on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL1, and copied to PSTATE.F on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

M[4], bit [4]

Execution state. Set to 0b0, the value of PSTATE.nRW, on taking an exception to EL1 from AArch64 state, and copied to PSTATE.nRW on executing an exception return operation in EL1.

M[4]	Meaning
0b0	AArch64 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch64 Exception level and selected Stack Pointer.

M[3:0]	Meaning
0b0000	EL0t.
0b0100	EL1t.
0b0101	EL1h.

Other values are reserved. If SPSR_EL1.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL1 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The bits in this field are interpreted as follows:

- M[3:2]: On an exception to EL1:
 - If the Effective value of [HCR_EL2](#).{NV, NV1} != {1,0} or the exception is not taken from EL1, then M[3:2] is set to the value of PSTATE.EL on taking an exception to EL1.
 - If the Effective value of [HCR_EL2](#).{NV, NV1} == {1,0} and the exception is not taken from EL1, then M[3:2] is set to 0b10.
 - M[3:2] is copied to PSTATE.EL on executing a legal exception return operation in EL1.
- M[1] is unused and is 0 for all non-reserved values.
- M[0] is set to the value of PSTATE.SP on taking an exception to EL1 and copied to PSTATE.SP on executing an exception return operation in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPSR_EL1

When [HCR_EL2](#).E2H is 1, without explicit synchronization, access from EL3 using the mnemonic SPSR_EL1 or SPSR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPSR_EL1

op0	op1	CRn	CRm	op2
------------	------------	------------	------------	------------

0b11	0b000	0b0100	0b0000	0b000
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x160];
    else
        X[t, 64] = SPSR_EL1;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            X[t, 64] = SPSR_EL2;
        else
            X[t, 64] = SPSR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = SPSR_EL1;

```

MSR SPSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x160] = X[t, 64];
    else
        SPSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            SPSR_EL2 = X[t, 64];
        else
            SPSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        SPSR_EL1 = X[t, 64];

```

MRS <Xt>, SPSR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x160];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = SPSR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = SPSR_EL1;
    else
        UNDEFINED;

```

MSR SPSR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x160] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPSR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SPSR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

MRS <Xt>, SPSR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = SPSR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPSR_EL2;

```

MSR SPSR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        SPSR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SPSR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPSR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SPSR_EL2, Saved Program Status Register (EL2)

The SPSR_EL2 characteristics are:

Purpose

Holds the saved process state when an exception is taken to EL2.

Configuration

AArch64 System register SPSR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [SPSR_hyp\[31:0\]](#).

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

SPSR_EL2 is a 64-bit register.

Field descriptions

When AArch32 is supported and exception taken from AArch32 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
N	Z	C	V	Q	IT[1:0]	DI	SSBS	PAN	SS	IL	GE				IT[7:2]				E	A	I	F	T	M[4]				M[3:0]			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

An exception return from EL2 using AArch64 makes SPSR_EL2 become UNKNOWN.

Bits [63:32]

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL2, and copied to PSTATE.N on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL2, and copied to PSTATE.Z on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL2, and copied to PSTATE.C on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL2, and copied to PSTATE.V on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Q, bit [27]

Overflow or saturation flag. Set to the value of PSTATE.Q on taking an exception to EL2, and copied to PSTATE.Q on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IT, bits [15:10, 26:25]

If-Then. Set to the value of PSTATE.IT on taking an exception to EL2, and copied to PSTATE.IT on executing an exception return operation in EL2.

SPSR_EL2.IT must contain a value that is valid for the instruction being returned to.

The IT field is split as follows:

- IT[1:0] is SPSR_EL2[26:25].
- IT[7:2] is SPSR_EL2[15:10].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIT, bit [24]**When FEAT_DIT is implemented:**

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL2, and copied to PSTATE.DIT on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [23]**When FEAT_SSBS is implemented:**

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL2, and copied to PSTATE.SSBS on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]

When FEAT_PAN is implemented:

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL2, and copied to PSTATE.PAN on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL2, and conditionally copied to PSTATE.SS on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL2, and copied to PSTATE.IL on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GE, bits [19:16]

Greater than or Equal flags. Set to the value of PSTATE.GE on taking an exception to EL2, and copied to PSTATE.GE on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E, bit [9]

Endianness. Set to the value of PSTATE.E on taking an exception to EL2, and copied to PSTATE.E on executing an exception return operation in EL2.

If the implementation does not support big-endian operation, SPSR_EL2.E is RES0. If the implementation does not support little-endian operation, SPSR_EL2.E is RES1. On executing an exception return operation in EL2, if the implementation does not support big-endian operation at the Exception level being returned to, SPSR_EL2.E is RES0, and if the implementation does not support little-endian operation at the Exception level being returned to, SPSR_EL2.E is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL2, and copied to PSTATE.A on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL2, and copied to PSTATE.I on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL2, and copied to PSTATE.F on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T, bit [5]

T32 Instruction set state. Set to the value of PSTATE.T on taking an exception to EL2, and copied to PSTATE.T on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[4], bit [4]

Execution state. Set to 0b1, the value of PSTATE.nRW, on taking an exception to EL2 from AArch32 state, and copied to PSTATE.nRW on executing an exception return operation in EL2.

M[4]	Meaning
0b1	AArch32 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch32 Mode. Set to the value of PSTATE.M[3:0] on taking an exception to EL2, and copied to PSTATE.M[3:0] on executing an exception return operation in EL2.

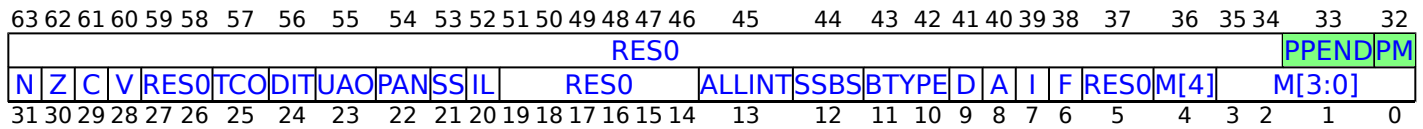
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

Other values are reserved. If SPSR_EL2.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL2 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When exception taken from AArch64 state:



An exception return from EL2 using AArch64 makes SPSR_EL2 become UNKNOWN.

Bits [63:34:32]

Reserved, RES0.

PPEND, bit [33]

When FEAT_SEBEP is implemented:

PMU exception pending bit. Set to the value of PSTATE.PPEND on taking an exception to EL2, and conditionally copied to PSTATE.PPEND on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PM, bit [32]

When FEAT_EBEP is implemented:

PMU exception mask bit. Set to the value of PSTATE.PM on taking an exception to EL2, and copied to PSTATE.PM on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL2, and copied to PSTATE.N on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL2, and copied to PSTATE.Z on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL2, and copied to PSTATE.C on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL2, and copied to PSTATE.V on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

TCO, bit [25]**When FEAT_MTE is implemented:**

Tag Check Override. Set to the value of PSTATE.TCO on taking an exception to EL2, and copied to PSTATE.TCO on executing an exception return operation in EL2.

When FEAT_MTE2 is not implemented, it is CONSTRAINED UNPREDICTABLE whether this field is RES0 or behaves as if FEAT_MTE2 is implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DIT, bit [24]**When FEAT_DIT is implemented:**

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL2, and copied to PSTATE.DIT on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UAO, bit [23]**When FEAT_UAO is implemented:**

User Access Override. Set to the value of PSTATE.UAO on taking an exception to EL2, and copied to PSTATE.UAO on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL2, and copied to PSTATE.PAN on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL2, and conditionally copied to PSTATE.SS on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL2, and copied to PSTATE.IL on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:14]

Reserved, RES0.

ALLINT, bit [13]**When FEAT_NMI is implemented:**

All IRQ or FIQ interrupts mask. Set to the value of PSTATE.ALLINT on taking an exception to EL2, and copied to PSTATE.ALLINT on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [12]**When FEAT_SSBS is implemented:**

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL2, and copied to PSTATE.SSBS on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BTYPE, bits [11:10]**When FEAT_BTI is implemented:**

Branch Type Indicator. Set to the value of PSTATE.BTYPE on taking an exception to EL2, and copied to PSTATE.BTYPE on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D, bit [9]

Debug exception mask. Set to the value of PSTATE.D on taking an exception to EL2, and copied to PSTATE.D on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

Error interrupt mask. Set to the value of PSTATE.A on taking an exception to EL2, and copied to PSTATE.A on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL2, and copied to PSTATE.I on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL2, and copied to PSTATE.F on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

M[4], bit [4]

Execution state. Set to 0b0, the value of PSTATE.nRW, on taking an exception to EL2 from AArch64 state, and copied to PSTATE.nRW on executing an exception return operation in EL2.

M[4]	Meaning
0b0	AArch64 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch64 Exception level and selected Stack Pointer.

M[3:0]	Meaning
0b0000	EL0t.
0b0100	EL1t.
0b0101	EL1h.
0b1000	EL2t.
0b1001	EL2h.

Other values are reserved. If SPSR_EL2.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL2 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The bits in this field are interpreted as follows:

- M[3:2] is set to the value of PSTATE.EL on taking an exception to EL2 and copied to PSTATE.EL on executing an exception return operation in EL2.
- M[1] is unused and is 0 for all non-reserved values.
- M[0] is set to the value of PSTATE.SP on taking an exception to EL2 and copied to PSTATE.SP on executing an exception return operation in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPSR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic SPSR_EL2 or SPSR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPSR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = SPSR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPSR_EL2;

```

MSR SPSR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        SPSR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SPSR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SPSR_EL2 = X[t, 64];

```

MRS <Xt>, SPSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x160];
    else
        X[t, 64] = SPSR_EL1;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            X[t, 64] = SPSR_EL2;
        else
            X[t, 64] = SPSR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = SPSR_EL1;

```

MSR SPSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x160] = X[t, 64];
    else
        SPSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            SPSR_EL2 = X[t, 64];
        else
            SPSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        SPSR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SPSR_EL3, Saved Program Status Register (EL3)

The SPSR_EL3 characteristics are:

Purpose

Holds the saved process state when an exception is taken to EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to SPSR_EL3 are UNDEFINED.

Attributes

SPSR_EL3 is a 64-bit register.

Field descriptions

When AArch32 is supported and exception taken from AArch32 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
N	Z	C	V	Q	IT[1:0]	DIT	SSBS	PAN	SS	IL	GE	IT[7:2]	E	A	I	F	T	M[4]	M[3:0]												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

An exception return from EL3 using AArch64 makes SPSR_EL1 become UNKNOWN.

Bits [63:32]

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL3, and copied to PSTATE.N on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL3, and copied to PSTATE.Z on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL3, and copied to PSTATE.C on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL3, and copied to PSTATE.V on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Q, bit [27]

Overflow or saturation flag. Set to the value of PSTATE.Q on taking an exception to EL3, and copied to PSTATE.Q on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IT, bits [15:10, 26:25]

If-Then. Set to the value of PSTATE.IT on taking an exception to EL3, and copied to PSTATE.IT on executing an exception return operation in EL3.

SPSR_EL1.IT must contain a value that is valid for the instruction being returned to.

The IT field is split as follows:

- IT[1:0] is SPSR_EL3[26:25].
- IT[7:2] is SPSR_EL3[15:10].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DIT, bit [24]

When FEAT_DIT is implemented:

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL3, and copied to PSTATE.DIT on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [23]

When FEAT_SSBS is implemented:

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL3, and copied to PSTATE.SSBS on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL3, and copied to PSTATE.PAN on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL3, and conditionally copied to PSTATE.SS on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL3, and copied to PSTATE.IL on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

GE, bits [19:16]

Greater than or Equal flags. Set to the value of PSTATE.GE on taking an exception to EL3, and copied to PSTATE.GE on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

E, bit [9]

Endianness. Set to the value of PSTATE.E on taking an exception to EL3, and copied to PSTATE.E on executing an exception return operation in EL3.

If the implementation does not support big-endian operation, SPSR_EL1.E is RES0. If the implementation does not support little-endian operation, SPSR_EL1.E is RES1. On executing an exception return operation in EL3, if the implementation does not support big-endian operation at the Exception level being returned to, SPSR_EL1.E is RES0, and if the implementation does not support little-endian operation at the Exception level being returned to, SPSR_EL1.E is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL3, and copied to PSTATE.A on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL3, and copied to PSTATE.I on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL3, and copied to PSTATE.F on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T, bit [5]

T32 Instruction set state. Set to the value of PSTATE.T on taking an exception to EL3, and copied to PSTATE.T on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[4], bit [4]

Execution state. Set to 0b1, the value of PSTATE.nRW, on taking an exception to EL3 from AArch32 state, and copied to PSTATE.nRW on executing an exception return operation in EL3.

M[4]	Meaning
0b1	AArch32 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch32 Mode. Set to the value of PSTATE.M[3:0] on taking an exception to EL3, and copied to PSTATE.M[3:0] on executing an exception return operation in EL3.

M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0110	Monitor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

Other values are reserved. If SPSR_EL1.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL3 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When exception taken from AArch64 state:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32					
RES0																															PPENDPM					
N	Z	C	V	RES0	TCODIT	UAO	PAN	SS	IL	RES0				ALLINT	SSBS	BTYPE	D	A	I	F	RES0	M[4]	M[3:0]													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					

An exception return from EL3 using AArch64 makes SPSR_EL1 become UNKNOWN.

Bits [63:34:32]

Reserved, RES0.

PPEND, bit [33]

When FEAT_SEBEP is implemented:

PMU exception pending bit. Set to the value of PSTATE.PPEND on taking an exception to EL3, and conditionally copied to PSTATE.PPEND on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PM, bit [32]

When FEAT_EBEP is implemented:

PMU exception mask bit. Set to the value of PSTATE.PM on taking an exception to EL3, and copied to PSTATE.PM on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

N, bit [31]

Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL3, and copied to PSTATE.N on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Z, bit [30]

Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL3, and copied to PSTATE.Z on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

C, bit [29]

Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL3, and copied to PSTATE.C on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

V, bit [28]

Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL3, and copied to PSTATE.V on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [27:26]

Reserved, RES0.

TCO, bit [25]**When FEAT_MTE is implemented:**

Tag Check Override. Set to the value of PSTATE.TCO on taking an exception to EL3, and copied to PSTATE.TCO on executing an exception return operation in EL3.

When FEAT_MTE2 is not implemented, it is CONSTRAINED UNPREDICTABLE whether this field is RES0 or behaves as if FEAT_MTE2FEAT_MTE is implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DIT, bit [24]**When FEAT_DIT is implemented:**

Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL3, and copied to PSTATE.DIT on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

UAO, bit [23]**When FEAT_UAO is implemented:**

User Access Override. Set to the value of PSTATE.UAO on taking an exception to EL3, and copied to PSTATE.UAO on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL3, and copied to PSTATE.PAN on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SS, bit [21]

Software Step. Set to the value of PSTATE.SS on taking an exception to EL3, and conditionally copied to PSTATE.SS on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IL, bit [20]

Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL3, and copied to PSTATE.IL on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [19:14]

Reserved, RES0.

ALLINT, bit [13]**When FEAT_NMI is implemented:**

All IRQ or FIQ interrupts mask. Set to the value of PSTATE.ALLINT on taking an exception to EL3, and copied to PSTATE.ALLINT on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SSBS, bit [12]**When FEAT_SSBS is implemented:**

Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL3, and copied to PSTATE.SSBS on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BTYPE, bits [11:10]**When FEAT_BTI is implemented:**

Branch Type Indicator. Set to the value of PSTATE.BTYPE on taking an exception to EL3, and copied to PSTATE.BTYPE on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D, bit [9]

Debug exception mask. Set to the value of PSTATE.D on taking an exception to EL3, and copied to PSTATE.D on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A, bit [8]

Error interrupt mask. Set to the value of PSTATE.A on taking an exception to EL3, and copied to PSTATE.A on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

I, bit [7]

IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL3, and copied to PSTATE.I on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

F, bit [6]

FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL3, and copied to PSTATE.F on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [5]

Reserved, RES0.

M[4], bit [4]

Execution state. Set to 0b0, the value of PSTATE.nRW, on taking an exception to EL3 from AArch64 state, and copied to PSTATE.nRW on executing an exception return operation in EL3.

M[4]	Meaning
0b0	AArch64 execution state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

M[3:0], bits [3:0]

AArch64 Exception level and selected Stack Pointer.

M[3:0]	Meaning
0b0000	EL0t.
0b0100	EL1t.
0b0101	EL1h.
0b1000	EL2t.
0b1001	EL2h.
0b1100	EL3t.
0b1101	EL3h.

Other values are reserved. If SPSR_EL1.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL3 is an illegal return event, as described in 'Illegal return events from AArch64 state'.

The bits in this field are interpreted as follows:

- M[3:2] is set to the value of PSTATE.EL on taking an exception to EL3 and copied to PSTATE.EL on executing an exception return operation in EL3.
- M[1] is unused and is 0 for all non-reserved values.
- M[0] is set to the value of PSTATE.SP on taking an exception to EL3 and copied to PSTATE.SP on executing an exception return operation in EL3.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing SPSR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SPSR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SPSR_EL3;
```

MSR SPSR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SPSR_EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SVCR, Streaming Vector Control Register

The SVCR characteristics are:

Purpose

Controls Streaming SVE mode and SME behavior.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SVCR are UNDEFINED.

Attributes

SVCR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														ZA	SM

Bits [63:2]

Reserved, RES0.

ZA, bit [1]

Enables SME ZA storage. If FEAT_SME2 is implemented, also enables SME2 ZT0 storage.

When this storage is disabled, execution of an instruction which can access it is trapped. The exception is reported using an ESR_ELx.{EC, SMTC} value of {0x1D, 0x3}.

The possible values of this bit are:

ZA	Meaning
0b0	SME ZA storage and, if implemented, ZT0 storage are invalid and not accessible. This control causes execution at any Exception level of instructions that can access this storage to be trapped.
0b1	SME ZA storage and, if implemented, ZT0 storage are valid and accessible. This control does not cause execution of any instructions to be trapped.

When a write to SVCR.ZA changes the value of PSTATE.ZA from 0 to 1, all implemented bits of the storage following are set to zero. applies:

- When changed from 0 to 1, all implemented bits of the storage are set to zero.
- When changed from 1 to 0, there is no observable change to the storage.

Changes to this field do not have an affect on the SVE vector and predicate registers and FPSR.

Changes to this field do not have an effect on the SVE vector and predicate registers and FPSR.

A direct or indirect read of ZA appears to occur in program order relative to a direct write of SVCR, and to MSR SVCRZA and MSR SVCRSMZA instructions, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

SM, bit [0]

Enables Streaming SVE mode.

When the PE is in Streaming SVE mode, the Streaming SVE vector length (SVL) applies to SVE instructions, and execution at any Exception level of an instruction which is illegal in that mode is trapped. The exception is reported using an ESR_ELx.{EC, SMTC} value of {0x1D, 0x1}.

When the PE is not in Streaming SVE mode, the SVE vector length (VL) applies to SVE instructions, and execution at any Exception level of an instruction which is only legal in that mode is trapped. The exception is reported using an ESR_ELx.{EC, SMTC} value of {0x1D, 0x2}.

The possible values of this bit are:

SM	Meaning
0b0	The PE is not in Streaming SVE mode.
0b1	The PE is in Streaming SVE mode.

When a write to SVCR.SM changes the value of PSTATE.SM, the following applies:

- When changed from 0 to 1, an entry to Streaming SVE mode is performed.
- When changed from 1 to 0, an exit from Streaming SVE mode is performed.
- All implemented bits of the SVE registers Z0-Z31, P0-P15, and FFR in the new mode are set to zero.
- [FPSR](#) in the new mode is set to 0x0000_0000_0800_009f, in which all cumulative status bits are set to 1.

Changes to this field do not have an affect on SME ZA **storage or, if implemented, ZT0** storage.

A direct or indirect read of SM appears to occur in program order relative to a direct write of SVCR, and to MSR SVCRSM and MSR SVCRSMZA instructions, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SVCR

SVCR is read/write and can be accessed from any Exception level.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, SVCR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b010

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.SMEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        else
            AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.SMEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = Zeros(62):PSTATE.<ZA,SM>;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = Zeros(62):PSTATE.<ZA,SM>;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = Zeros(62):PSTATE.<ZA,SM>;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = Zeros(62):PSTATE.<ZA,SM>;

```

MSR SVCR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b010

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.SMEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        else
            AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.SMEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SetPSTATE_SVCR(X[t, 32]);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SetPSTATE_SVCR(X[t, 32]);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SetPSTATE_SVCR(X[t, 32]);
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SetPSTATE_SVCR(X[t, 32]);

```

MSR SVCRSM, #<imm>

op0	op1	CRn	CRm	op2
0b00	0b011	0b0100	0b001x	0b011

MSR SVCRZA, #<imm>

op0	op1	CRn	CRm	op2
0b00	0b011	0b0100	0b010x	0b011

MSR SVCRCMZA, #<imm>

op0	op1	CRn	CRm	op2
0b00	0b011	0b0100	0b011x	0b011

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TCO, Tag Check Override

The TCO characteristics are:

Purpose

When FEAT_MTE is implemented, this register allows tag checks to be disabled globally.

When FEAT_MTE2 is not implemented, it is CONstrained UNpredictable whether this register is RES0 or behaves as if FEAT_MTE2FEAT_MTE is implemented.

Configuration

This register is present only when FEAT_MTE is implemented. Otherwise, direct accesses to TCO are UNDEFINED.

Attributes

TCO is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0						TCO		RES0																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:26]

Reserved, RES0.

TCO, bit [25]

Allows memory tag checks to be globally disabled.

TCO	Meaning
0b0	Loads and Stores are not affected by this control.
0b1	Loads and Stores are unchecked.

Bits [24:0]

Reserved, RES0.

Accessing TCO

For information about the operation of the MSR (immediate) accessor, see MSR (immediate).

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCO

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b111

```

if PSTATE.EL == EL0 then
    X[t, 64] = Zeros(38):PSTATE.TC0:Zeros(25);
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(38):PSTATE.TC0:Zeros(25);
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(38):PSTATE.TC0:Zeros(25);
elsif PSTATE.EL == EL3 then
    X[t, 64] = Zeros(38):PSTATE.TC0:Zeros(25);

```

MSR TC0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b111

```

if PSTATE.EL == EL0 then
    PSTATE.TCO = X[t, 64]<25>;
elsif PSTATE.EL == EL1 then
    PSTATE.TCO = X[t, 64]<25>;
elsif PSTATE.EL == EL2 then
    PSTATE.TCO = X[t, 64]<25>;
elsif PSTATE.EL == EL3 then
    PSTATE.TCO = X[t, 64]<25>;

```

MSR TCO, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b100

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TCR2_EL1, Extended Translation Control Register (EL1)

The TCR2_EL1 characteristics are:

Purpose

The control register for stage 1 of the EL1&0 translation regime.

Configuration

This register is present only when FEAT_TCR2 is implemented. Otherwise, direct accesses to TCR2_EL1 are UNDEFINED.

Attributes

TCR2_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																DisCH1	DisCH0	RES0	HAFT	PTTWI	RES0	D128	AIE	POE	E0	POE	PIE	PnCH			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Unless stated otherwise, all the bits in [TCR2_EL1](#), when they have the value 1, are permitted to be cached in a TLB.

Bits [63:16]

Reserved, RES0.

DisCH1, bit [15]

When FEAT_D128 is implemented and TCR2_EL1.D128 == 1:

Disable Contiguous Hint for Start Table for VARange 1.

DisCH1	Meaning
0b0	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 1 are not affected by this field.
0b1	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 1 are disabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DisCH0, bit [14]**When FEAT_D128 is implemented and TCR2_EL1.D128 == 1:**

Disable Contiguous Hint for Start Table for VARange 0.

DisCH0	Meaning
0b0	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 0 are not affected by this field.
0b1	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 0 are disabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [13:12]

Reserved, RES0.

HAFT, bit [11]**When FEAT_HAFT is implemented:**

Hardware managed Access Flag for Tables.

Enables the Hardware managed Access Flag for Tables.

HAFT	Meaning
0b0	Hardware managed Access Flag for Tables is disabled.
0b1	Hardware managed Access Flag for Tables is enabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PTTWI, bit [10]**When FEAT_THE is implemented:**

Permit Translation table walk Incoherence.

Permits RCWS instructions to have Reduced Coherence property.

PTTWI	Meaning
0b0	Write accesses generated by RCWS at EL1&0 do not have the Reduced Coherence property.
0b1	Write accesses generated by RCWS at EL1&0 have the Reduced Coherence property if HCRX_EL2 .PTTWI is 1.

This bit is permitted to be built as a read-only bit with a fixed value of 0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [9:6]

Reserved, RES0.

D128, bit [5]

When FEAT_D128 is implemented:

Enable 128-bit Page Table Descriptors.

Enables VMSAv9-128 translation system for the Stage 1 EL1&0 Translation Process.

D128	Meaning
0b0	Translation system follows VMSA-64 translation process.
0b1	Translation system follows VMSAv9-128 translation process.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AIE, bit [4]

When FEAT_AIE is implemented:

Enable Attribute Indexing Extension. Control for Attribute Indexing Extension for Stage 1 EL1&0 Translation Process.

AIE	Meaning
0b0	Attribute Indexing Extension Disabled.
0b1	Attribute Indexing Extension Enabled.

This field is RES1 when [TCR2_EL1.D128](#) is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

POE, bit [3]**When FEAT_S1POE is implemented:**

POE. Controls setting of permission overlay for EL1 accesses in stage 1 of the EL1&0 translation regime.

POE	Meaning
0b0	Permission overlay disabled for EL1 access in stage 1 of EL1&0 translation regime.
0b1	Permission overlay enabled for EL1 access in stage 1 of EL1&0 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E0POE, bit [2]**When FEAT_S1POE is implemented:**

E0POE. controls setting of permission overlay in stage 1 of the EL1 translation regime.

E0POE	Meaning
0b0	Permission overlay disabled for EL0 access in stage 1 of EL1&0 translation regime.
0b1	Permission overlay enabled for EL0 access in stage 1 of EL1&0 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PIE, bit [1]**When FEAT_S1PIE is implemented:**

Select Permission Model. Controls setting of indirect permission model in Stage 1 EL1 Translation Process.

PIE	Meaning
0b0	Direct permission model.
0b1	Indirect permission model.

This field is RES1 when [TCR2_EL1.D128](#) is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PnCH, bit [0]**When FEAT_TTE is implemented:**

Protected attribute enable. Indicates use of bit[52] of the stage 1 translation table entry.

PnCH	Meaning
0b0	Bit[52] of each stage 1 translation table entry does not indicate protected attribute.
0b1	Bit[52] of each stage 1 translation table entry indicates protected attribute.

This field is RES0 when [TCR2_EL1.D128](#) is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL2 is not implemented and EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TCR2_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.TCR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            X[t, 64] = NVMem[0x270];
        else
            X[t, 64] = TCR2_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            X[t, 64] = TCR2_EL2;
        else
            X[t, 64] = TCR2_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = TCR2_EL1;

```

MSR TCR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.TCR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x270] = X[t, 64];
        else
            TCR2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HCR_EL2.E2H == '1' then
            TCR2_EL2 = X[t, 64];
        else
            TCR2_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        TCR2_EL1 = X[t, 64];

```

MRS <Xt>, TCR2_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x270];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = MAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR_EL1;
    else
        UNDEFINED;

```

MSR TCR2_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x270] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

no old file

htmldiff from-

(new)

TCR2_EL2, Extended Translation Control Register (EL2)

The TCR2_EL2 characteristics are:

Purpose

The control register for stage 1 of the EL2&0 translation regime.

Configuration

This register is present only when FEAT_TCR2 is implemented. Otherwise, direct accesses to TCR2_EL2 are UNDEFINED.

Attributes

TCR2_EL2 is a 64-bit register.

Field descriptions

When HCR_EL2.E2H == 0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
																RES0																			
RES0														AMEC1		AMEC0		HAFT		PTTWI		RES0		D128		AIE		POE		RES0		PIE		PnCH	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Unless stated otherwise, all the bits in [TCR2_EL2](#), when they have the value 1, are permitted to be cached in a TLB.

Bits [63:14]

Reserved, RES0.

AMEC1, bit [13]

When FEAT_MEC is implemented and FEAT_VHE is implemented:

This field controls the enabling of the Alternate MECID translations for the EL2&0 TTBR1 translation regime.

TCR2_EL2.AMEC1 is provided to enable the safe update of [TTBR_EL2](#) and [MECID_A_EL2](#), by disabling access and speculation to AMEC == 1 Block or Page descriptors during the update.

AMEC1	Meaning
0b0	Use of a Block or Page descriptor containing AMEC == 1 generates a Translation fault.
0b1	Accesses translated by a Block or Page descriptor containing AMEC == 1 are associated with the MECID configured in MECID_A1_EL2 .

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When IsCurrentSecurityState(SS_Secure), access to this field is **RES0**.
- When IsCurrentSecurityState(SS_NonSecure), access to this field is **RES0**.
- When SCTLR2_EL2.MECRL != 0, access to this field is **RES1**.

Otherwise:

Reserved, RES0.

AMEC0, bit [12]**When FEAT_MEC is implemented:**

This field controls the enabling of the Alternate MECID translations for the EL2 and EL2&0 TTBR0 translation regimes.

TCR2_EL2.AMEC0 is provided to enable the safe update of [TTBR_EL2](#) and [MECID_A_EL2](#), by disabling access and speculation to AMEC=1 Block or Page descriptors during the update.

AMEC0	Meaning
0b0	Use of a Block or Page descriptor containing AMEC == 1 generates a Translation fault.
0b1	Accesses translated by a Block or Page descriptor containing AMEC == 1 are associated with the MECID configured in MECID_A0_EL2 .

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When IsCurrentSecurityState(SS_Secure), access to this field is **RES0**.
- When IsCurrentSecurityState(SS_NonSecure), access to this field is **RES0**.
- When SCTLR2_EL2.MECRL != 0, access to this field is **RES1**.

Otherwise:

Reserved, RES0.

HAFT, bit [11]**When FEAT_HAFT is implemented:**

Hardware managed Access Flag for Tables.

Enables the Hardware managed Access Flag for Tables.

HAFT	Meaning
0b0	Hardware managed Access Flag for Tables is disabled.
0b1	Hardware managed Access Flag for Tables is enabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PTTWI, bit [10]**When FEAT_THE is implemented:**

Permit Translation table walk Incoherence.

Permits RCWS instructions to have Reduced Coherence property.

PTTWI	Meaning
0b0	Write accesses generated by RCWS at EL2 or EL2&0 do not have the Reduced Coherence property.
0b1	Write accesses generated by RCWS at EL2 or EL2&0 have the Reduced Coherence property.

This bit is permitted to be built as a read-only bit with a fixed value of 0.

This field is ignored by the PE and treated as zero when [SCR_EL3.TCR2En](#) == 0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [9:6]

Reserved, RES0.

D128, bit [5]**When FEAT_D128 is implemented:**

Enable 128-bit Page Table Descriptors.

Enables VMSAv9-128 translation system for the Stage 1 EL2 Translation Process.

D128	Meaning
0b0	Translation system follows VMSA-64 translation process.
0b1	Translation system follows VMSAv9-128 translation process.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AIE, bit [4]**When FEAT_AIE is implemented:**

Enable Attribute Indexing Extension. Control for Attribute Indexing Extension for Stage 1 EL2 Translation Process.

AIE	Meaning
0b0	Attribute Indexing Extension Disabled.
0b1	Attribute Indexing Extension Enabled.

This field is RES1 when TCR2_EL2.D128 is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

POE, bit [3]

When FEAT_S1POE is implemented:

POE. Controls setting of permission overlay for EL2 accesses in stage 1 of the EL2 translation regime.

POE	Meaning
0b0	Permission overlay disabled for EL2 access in stage 1 of EL2 translation regime.
0b1	Permission overlay enabled for EL2 access in stage 1 of EL2 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [2]

Reserved, RES0.

PIE, bit [1]

When FEAT_S1PIE is implemented:

Select Permission Model. Controls setting of indirect permission model in Stage 1 EL2 Translation Process.

PIE	Meaning
0b0	Direct permission model.
0b1	Indirect permission model.

This field is RES1 when TCR2_EL2.D128 is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PnCH, bit [0]**When FEAT_THE is implemented:**

Protected attribute enable. Indicates use of bit[52] of the stage 1 translation table entry.

PnCH	Meaning
0b0	Bit[52] of each stage 1 translation table entry does not indicate protected attribute.
0b1	Bit[52] of each stage 1 translation table entry indicates protected attribute.

This field is RES0 when TCR2_EL2.D128 is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When HCR_EL2.E2H == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
RES0																DisCH1	DisCH0	RES0	AMEC0	HAFT	PTTW	SKL1	SKL0	D128	AIE	POE	E0	POE	PIE	PnCH	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Unless stated otherwise, all the bits in [TCR2_EL2](#), when they have the value 1, are permitted to be cached in a TLB.

Bits [63:16]

Reserved, RES0.

DisCH1, bit [15]**When FEAT_D128 is implemented and TCR2_EL2.D128 == 1:**

Disable Contiguous Hint for Start Table for VARange 1.

DisCH1	Meaning
0b0	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 1 are not affected by this field.
0b1	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 1 are disabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DisCH0, bit [14]**When FEAT_D128 is implemented and TCR2_EL2.D128 == 1:**

Disable Contiguous Hint for Start Table for VARange 0.

DisCH0	Meaning
0b0	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 0 are not affected by this field.
0b1	Contiguous Hint of Block or Page descriptors of the Start Table for VARange 0 are disabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [13]

Reserved, RES0.

AMEC0, bit [12]**When FEAT_MEC is implemented:**

This field controls the enabling of the Alternate MECID translations for the EL2 and EL2&0 TTBR0 translation regimes.

TCR2_EL2.AMEC0 is provided to enable the safe update of [TTBR_EL2](#) and [MECID_A_EL2](#), by disabling access and speculation to AMEC=1 Block or Page descriptors during the update.

AMEC0	Meaning
0b0	Use of a Block or Page descriptor containing AMEC == 1 generates a Translation fault.
0b1	Accesses translated by a Block or Page descriptor containing AMEC == 1 are associated with the MECID configured in MECID_A0_EL2 .

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When IsCurrentSecurityState(SS_Secure), access to this field is **RES0**.
- When IsCurrentSecurityState(SS_NonSecure), access to this field is **RES0**.
- When SCTLR2_EL2.MECRL != 0, access to this field is **RES1**.

Otherwise:

Reserved, RES0.

HAFT, bit [11]**When FEAT_HAFT is implemented:**

Hardware managed Access Flag for Tables.

Enables the Hardware managed Access Flag for Tables.

HAFT	Meaning
0b0	Hardware managed Access Flag for Tables is disabled.
0b1	Hardware managed Access Flag for Tables is enabled.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PTTWI, bit [10]

When FEAT_THE is implemented:

Permit Translation table walk Incoherence.

Permits RCWS instructions to have Reduced Coherence property.

PTTWI	Meaning
0b0	Write accesses generated by RCWS do not have the Reduced Coherence property.
0b1	Write accesses generated by RCWS have the Reduced Coherence property.

This bit is permitted to be built as a read-only bit with a fixed value of 0.

This field is ignored by the PE and treated as zero when [SCR_EL3.TCR2En](#) == 0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SKL1, bits [9:8]

When FEAT_D128 is implemented:

Skip Level associated with translation table walks using [TTBR1_EL2](#).

This determines the number of levels to be skipped in the regular start level of the Stage 1 EL2&0 translation table walks using [TTBR1_EL2](#).

SKL1	Meaning
0b00	Skip 0 level in the regular start level.
0b01	Skip 1 level in the regular start level.
0b10	Skip 2 levels in the regular start level.
0b11	Skip 3 levels in the regular start level.

This field is IGNORED when [TCR2_EL2.D128](#) is 0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.

- Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SKL0, bits [7:6]**When FEAT_D128 is implemented:**

Skip Level associated with translation table walks using [TTBR0_EL2](#).

This determines the number of levels to be skipped in the regular start level of the Stage 1 EL2&0 translation table walks using [TTBR0_EL2](#).

SKL0	Meaning
0b00	Skip 0 level in the regular start level.
0b01	Skip 1 level in the regular start level.
0b10	Skip 2 levels in the regular start level.
0b11	Skip 3 levels in the regular start level.

This field is IGNORED when [TCR2_EL2.D128](#) is 0.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

D128, bit [5]**When FEAT_D128 is implemented:**

Enable 128-bit Page Table Descriptors.

Enables VMSAv9-128 translation system for the Stage 1 EL2&0 Translation Process.

D128	Meaning
0b0	Translation system follows VMSA-64 translation process.
0b1	Translation system follows VMSAv9-128 translation process.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AIE, bit [4]**When FEAT_AIE is implemented:**

Enable Attribute Indexing Extension. Control for Attribute Indexing Extension for Stage 1 EL2&0 Translation Process.

AIE	Meaning
0b0	Attribute Indexing Extension Disabled.
0b1	Attribute Indexing Extension Enabled.

This field is RES1 when [TCR2_EL2.D128](#) is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

POE, bit [3]

When FEAT_S1POE is implemented:

POE. Controls setting of permission overlay for EL2 accesses in stage 1 of the EL2&0 translation regime.

POE	Meaning
0b0	Permission overlay disabled for EL2 access in stage 1 of EL2&0 translation regime.
0b1	Permission overlay enabled for EL2 access in stage 1 of EL2&0 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E0POE, bit [2]

When FEAT_S1POE is implemented:

E0POE. controls setting of permission overlay in stage 1 of the EL2 translation regime.

E0POE	Meaning
0b0	Permission overlay disabled for EL0 access in stage 1 of EL2&0 translation regime.
0b1	Permission overlay enabled for EL0 access in stage 1 of EL2&0 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PIE, bit [1]**When FEAT_S1PIE is implemented:**

Select Permission Model. Controls setting of indirect permission model in Stage 1 EL2 Translation Process.

PIE	Meaning
0b0	Direct permission model.
0b1	Indirect permission model.

This field is RES1 when [TCR2_EL2](#).D128 is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PnCH, bit [0]**When FEAT_THE is implemented:**

Protected attribute enable. Indicates use of bit[52] of the stage 1 translation table entry.

PnCH	Meaning
0b0	Bit[52] of each stage 1 translation table entry does not indicate protected attribute.
0b1	Bit[52] of each stage 1 translation table entry indicate protected attribute.

This field is RES1 when [TCR2_EL2](#).D128 is set.

The reset behavior of this field is:

- On a Warm reset:
 - When EL3 is not implemented, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TCR2_EL2

When FEAT_VHE is implemented, and [HCR_EL2](#).E2H is 1, without explicit synchronization, accesses from EL2 using the register name TCR2_EL2 or TCR2_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TCR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR2_EL2;

```

MSR TCR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TCR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TCR2_EL2 = X[t, 64];

```

MRS <Xt>, TCR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.TCR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x270];
    else
        X[t, 64] = TCR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = TCR2_EL2;
    else
        X[t, 64] = TCR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR2_EL1;

```

MSR TCR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.TCR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x270] = X[t, 64];
    else
        TCR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TCR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.TCR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        TCR2_EL2 = X[t, 64];
    else
        TCR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TCR2_EL1 = X[t, 64];

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

The TCR EL1 characteristics are:

Purpose

The control register for stage 1 of the EL1&0 translation regime.

Configuration

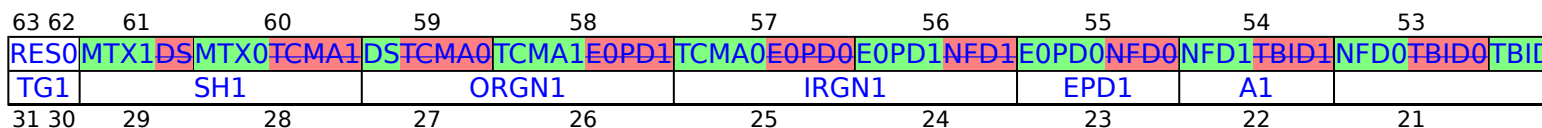
AArch64 System register TCR EL1 bits [31:0] are architecturally mapped to AArch32 System register [TTBCR\[31:0\]](#).

AArch64 System register TCR EL1 bits [63:32] are architecturally mapped to AArch32 System register [TTBCR2\[31:0\]](#).

Attributes

TCR EL1 is a 64-bit register.

Field descriptions



Any of the bits in TCR_EL1, other than the A1 bit and the EPDx bits when they have the value 1, are permitted to be cached in a TLB.

Bits [63:6260]

Reserved, RES0.

MTX1, bit [61]

When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:

Extended memory tag checking.

This field controls address generation and tag checking when EL0 and EL1 are using AArch64 where the data address would be translated by tables pointed to by [TTBR1_EL1](#).

This control has an effect regardless of whether stage 1 of the EL1&0 translation regime is enabled or not.

MTX1	Meaning
0b0	This control has no effect on the PE.
0b1	<p>Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply:</p> <ul style="list-style-type: none"> Bits[59:56] are treated as 0b1111 when checking if the address is out of range. If FEAT_PAAuth is implemented, bits[59:56] are not part of the PAC field. A Canonical Tag Check operation is performed on Tag Checked memory accesses to a Canonically Tagged memory location.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MTX0, bit [60]

When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:

Extended memory tag checking.

This field controls address generation and tag checking when EL0 and EL1 are using AArch64 where the data address would be translated by tables pointed to by TTBR0_EL1.

This control has an effect regardless of whether stage 1 of the EL1&0 translation regime is enabled or not.

MTX0	Meaning
0b0	This control has no effect on the PE.
0b1	Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply: <ul style="list-style-type: none">Bits[59:56] are treated as 0b0000 when checking if the address is out of range.If FEAT_PAuth is implemented, bits[59:56] are not part of the PAC field.A Canonical Tag Check operation is performed on Tag Checked memory accesses to a Canonically Tagged memory location.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DS, bit [59]

When FEAT_LPA2 is implemented:

This field affects whether a 52-bit output address addressing can when be using described 4KB by and the 16KB translation tables granules in stage 1 of the 4KB EL1&0 or 16KB translation granules regime.

DS	Meaning
0b0	<p>Bits[49:48] of translation descriptors are RES0.</p> <p>Bits[9:8] in Block and Page descriptors encode shareability information in the SH[1:0] field. Bits[9:8] in Table descriptors are ignored by hardware.</p> <p>The minimum value of the TCR_EL1.{T0SZ, T1SZ} fields is 16. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>Output address[51:48] is 0b0000.</p>
0b1	<p>Bits[49:48] of translation descriptors hold output address[49:48].</p> <p>Bits[9:8] of Translation table descriptors hold output address[51:50].</p> <p>The shareability information of Block and Page descriptors for cacheable locations is determined by:</p> <ul style="list-style-type: none"> • TCR_EL1.SH0 if the VA is translated using tables pointed to by TTBR0_EL1. • TCR_EL1.SH1 if the VA is translated using tables pointed to by TTBR1_EL1. <p>The minimum value of the TCR_EL1.{T0SZ, T1SZ} fields is 12. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>All calculations of the stage 1 base address are modified for tables of fewer than 8 entries so that the table is aligned to 64 bytes.</p> <p>Bits[5:2] of TTBR0_EL1 or TTBR1_EL1 are used to hold bits[51:48] of the output address in all cases.</p> <hr/> <p>Note</p> <p>As FEAT_LVA must be implemented if TCR_EL1.DS == 1, the minimum value of the TCR_EL1.{T0SZ, T1SZ} fields is 12, as determined by that extension.</p> <hr/> <p>For the TLBI Range instructions affecting VA, the format of the argument is changed so that bits[36:0] hold BaseADDR[52:16]. For the 4KB translation granule, bits[15:12] of BaseADDR are treated as 0b0000. For the 16KB translation granule, bits[15:14] of BaseADDR are treated as 0b00.</p> <hr/> <p>Note</p> <p>This forces alignment of the ranges used by the TLBI range instructions.</p>

This field is RES0 for a 64KB translation granule.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCMA1, bit [58]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL1, and at EL0 if [HCR_EL2](#).{E2H,TGE}!={1,1}, when address[59:55] = 0b11111.

TCMA1	Meaning
0b0	This control has no effect on the generation of Unchecked accesses at EL1 or EL0.
0b1	All accesses at EL1 and EL0 are Unchecked.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCMA0, bit [57]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL1, and at EL0 if [HCR_EL2](#).{E2H,TGE}!={1,1}, when address[59:55] = 0b00000.

TCMA0	Meaning
0b0	This control has no effect on the generation of Unchecked accesses at EL1 or EL0.
0b1	All accesses at EL1 and EL0 are Unchecked.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EOPD1, bit [56]

When FEAT_EOPD is implemented:

Faulting control for Unprivileged access to any address translated by [TTBR1_EL1](#).

EOPD1	Meaning
0b0	Unprivileged access to any address translated by TTBR1_EL1 will not generate a fault by this mechanism.
0b1	Unprivileged access to any address translated by TTBR1_EL1 will generate a level 0 Translation fault.

Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EOPD0, bit [55]**When FEAT_EOPD is implemented:**

Faulting control for Unprivileged access to any address translated by [TTBR0_EL1](#).

EOPD0	Meaning
0b0	Unprivileged access to any address translated by TTBR0_EL1 will not generate a fault by this mechanism.
0b1	Unprivileged access to any address translated by TTBR0_EL1 will generate a level 0 Translation fault.

Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NFD1, bit [54]**When FEAT_SVE is implemented or FEAT_TME is implemented:**

Non-fault translation table walk disable for stage 1 translations using [TTBR1_EL1](#).

This bit controls whether to perform a stage 1 translation table walk in response to a non-fault unprivileged access for a virtual address that is translated using [TTBR1_EL1](#).

If SVE is implemented, the affected access types include:

- All accesses due to an SVE non-fault contiguous load instruction.
- Accesses due to an SVE first-fault gather load instruction that are not for the First active element. Accesses due to an SVE first-fault contiguous load instruction are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

For more information, see 'The Scalable Vector Extension (SVE)'.

If FEAT_TME is implemented, the affected access types include all accesses generated by a load or store instruction in Transactional state.

NFD1	Meaning
0b0	Does not disable stage 1 translation table walks using TTBR1_EL1 .
0b1	A TLB miss on a virtual address that is translated using TTBR1_EL1 due to the specified access types causes the access to fail without taking an exception. No stage 1 translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NFD0, bit [53]**When FEAT_SVE is implemented or FEAT_TME is implemented:**

Non-fault translation table walk disable for stage 1 translations using [TTBR0_EL1](#).

This bit controls whether to perform a stage 1 translation table walk in response to a non-fault unprivileged access for a virtual address that is translated using [TTBR0_EL1](#).

If SVE is implemented, the affected access types include:

- All accesses due to an SVE non-fault contiguous load instruction.
- Accesses due to an SVE first-fault gather load instruction that are not for the First active element. Accesses due to an SVE first-fault contiguous load instruction are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

For more information, see 'The Scalable Vector Extension (SVE)'.

If FEAT_TME is implemented, the affected access types include all accesses generated by a load or store instruction in Transactional state.

NFD0	Meaning
0b0	Does not disable stage 1 translation table walks using TTBR0_EL1 .
0b1	A TLB miss on a virtual address that is translated using TTBR0_EL1 due to the specified access types causes the access to fail without taking an exception. No stage 1 translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID1, bit [52]**When FEAT_PAuth is implemented:**

Controls the use of the top byte of instruction addresses for address matching.

For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.

For more information, see 'Address tagging in AArch64 state'.

TBID1	Meaning
0b0	TCR_EL1.TBI1 applies to Instruction and Data accesses.
0b1	TCR_EL1.TBI1 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR1_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID0, bit [51]**When FEAT_PAuth is implemented:**

Controls the use of the top byte of instruction addresses for address matching.

For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.

For more information, see 'Address tagging in AArch64 state'.

TBID0	Meaning
0b0	TCR_EL1.TBI0 applies to Instruction and Data accesses.
0b1	TCR_EL1.TBI0 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL1](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU162, bit [50]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

HWU162	Meaning
0b0	For translations using TTBR1_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU161, bit [49]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

HWU161	Meaning
0b0	For translations using TTBR1_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU160, bit [48]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

HWU160	Meaning
0b0	For translations using TTBR1_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU159, bit [47]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

HWU159	Meaning
0b0	For translations using TTBR1_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU062, bit [46]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU062	Meaning
0b0	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU061, bit [45]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU061	Meaning
0b0	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU060, bit [44]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU060	Meaning
0b0	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU059, bit [43]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU059	Meaning
0b0	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD1, bit [42]**When FEAT_HPDS is implemented:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR1_EL1](#).

HPD1	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD0, bit [41]

When FEAT_HPDS is implemented:

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR0_EL1](#).

HPD0	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HD, bit [40]

When FEAT_HAFDBS is implemented:

Hardware management of dirty state in stage 1 translations from EL0 and EL1.

HD	Meaning
0b0	Stage 1 hardware management of dirty state disabled.
0b1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HA, bit [39]

When FEAT_HAFDBS is implemented:

Hardware Access flag update in stage 1 translations from EL0 and EL1.

HA	Meaning
0b0	Stage 1 Access flag update disabled.
0b1	Stage 1 Access flag update enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBI1, bit [38]

Top Byte ignored. Indicates whether the top byte of an address is used for address match for the [TTBR1_EL1](#) region, or ignored and used for tagged addresses.

TBI1	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by [TTBR1_EL1](#). It has an effect whether the EL1&0 translation regime is enabled or not.

If FEAT_PAAuth is implemented and TCR_EL1.TBID1 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI1 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TBI0, bit [37]

Top Byte ignored. Indicates whether the top byte of an address is used for address match for the [TTBR0_EL1](#) region, or ignored and used for tagged addresses.

TBI0	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL1](#). It has an effect whether the EL1&0 translation regime is enabled or not.

If FEAT_PAAuth is implemented and TCR_EL1.TBID0 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI0 is 1 and bit [55] of the target address to be stored to the PC is 0, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AS, bit [36]

ASID Size.

AS	Meaning
0b0	8 bit - the upper 8 bits of TTBR0_EL1 and TTBR1_EL1 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros for when used for allocation and matching entries in the TLB.
0b1	16 bit - the upper 16 bits of TTBR0_EL1 and TTBR1_EL1 are used for allocation and matching in the TLB.

If the implementation has only 8 bits of ASID, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [35]

Reserved, RES0.

IPS, bits [34:32]

Intermediate Physical Address Size.

IPS	Meaning	Applies when
0b000	32 bits, 4GB.	
0b001	36 bits, 64GB.	
0b010	40 bits, 1TB.	
0b011	42 bits, 4TB.	
0b100	44 bits, 16TB.	
0b101	48 bits, 256TB.	
0b110	52 bits, 4PB.	
0b111	56 bits, 64PB.	When FEAT_D128 is implemented

All other values are reserved.

The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

If the translation granule is not 64KB and FEAT_LPA2 is not implemented, the value 0b110 is treated as reserved.

It is IMPLEMENTATION DEFINED whether an implementation that does not implement FEAT_LPA supports setting the value of 0b110 for the 64KB translation granule size or whether setting this value behaves as the 0b101 encoding.

If in an implementation that supports 52-bit PAs, if the value of this field is not ID_AA64MMFR0_EL1.PARange is 0b0110, and the value of this field is not 0b110 or a value treated as 0b110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL1 are 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG1, bits [31:30]

Granule size for the TTBR1_EL1.

TG1	Meaning
0b01	16KB.
0b10	4KB.
0b11	64KB.

Other values are reserved.

If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH1, bits [29:28]

Shareability attribute for memory associated with translation table walks using TTBR1_EL1.

SH1	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN1, bits [27:26]

Outer cacheability attribute for memory associated with translation table walks using [TTBR1_EL1](#).

ORGN1	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN1, bits [25:24]

Inner cacheability attribute for memory associated with translation table walks using [TTBR1_EL1](#).

IRGN1	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EPD1, bit [23]

Translation table walk disable for translations using [TTBR1_EL1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1_EL1](#). The encoding of this bit is:

EPD1	Meaning
0b0	Perform translation table walks using TTBR1_EL1 .
0b1	A TLB miss on an address that is translated using TTBR1_EL1 generates a Translation fault. No translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A1, bit [22]

Selects whether [TTBR0_EL1](#) or [TTBR1_EL1](#) defines the ASID. The encoding of this bit is:

A1	Meaning
0b0	TTBR0_EL1 .ASID defines the ASID.
0b1	TTBR1_EL1 .ASID defines the ASID.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T1SZ, bits [21:16]

The size offset of the memory region addressed by [TTBR1_EL1](#). The region size is $2^{(64-T1SZ)}$ bytes.

The maximum and minimum possible values for T1SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG0, bits [15:14]

Granule size for the [TTBR0_EL1](#).

TG0	Meaning
0b00	4KB
0b01	64KB
0b10	16KB

Other values are reserved.

If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

SH0	Meaning
0b00	Non-shareable
0b10	Outer Shareable
0b11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

ORGN0	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

IRGN0	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EPD0, bit [7]

Translation table walk disable for translations using [TTBR0_EL1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR0_EL1](#). The encoding of this bit is:

EPD0	Meaning
0b0	Perform translation table walks using TTBR0_EL1 .
0b1	A TLB miss on an address that is translated using TTBR0_EL1 generates a Translation fault. No translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL1](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TCR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TCR_EL1 or TCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x120];
    else
        X[t, 64] = TCR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TCR_EL2;
    else
        X[t, 64] = TCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR_EL1;

```

MSR TCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x120] = X[t, 64];
    else
        TCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TCR_EL2 = X[t, 64];
    else
        TCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TCR_EL1 = X[t, 64];

```

MRS <Xt>, TCR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x120];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = TCR_EL1;
    else
        UNDEFINED;

```

MSR TCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x120] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TCR_EL2 characteristics are:

Purpose

The control register for stage 1 of the EL2, or EL2&0, translation regime:

- When the Effective value of [HCR_EL2.E2H](#) is 0, this register controls stage 1 of the EL2 translation regime, that supports a single VA range, translated using [TTBR0_EL2](#).
- When the value of [HCR_EL2.E2H](#) is 1, this register controls stage 1 of the EL2&0 translation regime, that supports both:
 - A lower VA range, translated using [TTBR0_EL2](#).
 - A higher VA range, translated using [TTBR1_EL2](#).

Configuration

AArch64 System register TCR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HTCR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

TCR_EL2 is a 64-bit register.

Field descriptions

When HCR_EL2.E2H == 0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34			
RES0																																M
RES1	TCMA	TBID	HWU62	HWU61	HWU60	HWU59	HPD	RES1	HD	HA	TBI	RES0	PS	TG0	SH0	ORGN0	IRGN0	RES0	TOS													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2			

Any of the bits in TCR_EL2, other than the A1 bit and the EPDx bits when they have the value 1, are permitted to be cached in a TLB.

Bits [63:34~~33~~]

Reserved, RES0.

MTX, bit [33]

When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:

Extended memory tag checking.

This field controls address generation and tag checking when EL2 is using AArch64 where the data address would be translated by tables pointed to by [TTBRO EL2](#).

This control has an effect regardless of whether stage 1 of the EL2 translation regime is enabled or not.

MTX	Meaning
0b0	This control has no effect on the PE.
0b1	Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply: <ul style="list-style-type: none"> Bits[59:56] are treated as 0b0000 when checking if the address is out of range. If FEAT_PAAuth is implemented, bits[59:56] are not part of the PAC field. A Canonical Tag Check operation is performed on Tag Checked memory accesses to a Canonically Tagged memory location.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DS, bit [32]

When FEAT_LPA2 is implemented:

This field affects whether a 52-bit output address addressing can when be using described 4KB by and the 16KB translation tables granules in stage 1 of the 4KBEL2 or 16KB translation granules regime.

DS	Meaning
0b0	<p>Bits[49:48] of translation descriptors are RES0.</p> <p>Bits[9:8] in Block and Page descriptors encode shareability information in the SH[1:0] field. Bits[9:8] in table descriptors are ignored by hardware.</p> <p>The minimum value of TCR_EL2.T0SZ is 16. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p>
0b1	<p>Output address[51:48] is 0b0000.</p> <p>Bits[49:48] of translation descriptors hold output address[49:48].</p> <p>Bits[9:8] of Translation table descriptors hold output address[51:50].</p> <p>The shareability information of Block and Page descriptors for cacheable locations is determined by TCR_EL2.SH0.</p> <p>The minimum value of TCR_EL2.T0SZ is 12. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>All calculations of the stage 1 base address are modified for tables of fewer than 8 entries so that the table is aligned to 64 bytes.</p> <p>Bits[5:2] of TTBRO_EL2 are used to hold bits[51:48] of the output address in all cases.</p>

Note

As FEAT_LVA must be implemented if TCR_EL2.DS == 1, the minimum value of the TCR_EL2.T0SZ field is 12, as determined by that extension.

For the TLBI Range instructions affecting VA, the format of the argument is changed so that bits[36:0] hold BaseADDR[52:16]. For the 4KB translation granule, bits[15:12] of BaseADDR are treated as 0b0000. For the 16KB translation granule, bits[15:14] of BaseADDR are treated as 0b00.

Note

This forces alignment of the ranges used by the TLBI range instructions.

This field is RES0 for a 64KB translation granule.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [31]

Reserved, RES1.

TCMA, bit [30]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL2 when address [59:56] = 0b0000.

TCMA	Meaning
0b0	This control has no effect on the generation of Unchecked accesses.
0b1	All accesses are Unchecked.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID, bit [29]

When FEAT_PAuth is implemented:

Controls the use of the top byte of instruction addresses for address matching.

For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.

For more information, see 'Address tagging in AArch64 state'.

TBID	Meaning
0b0	TCR_EL2.TBI applies to Instruction and Data accesses.
0b1	TCR_EL2.TBI applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL2](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU62, bit [28]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry.

HWU62	Meaning
0b0	Bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU61, bit [27]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry.

HWU61	Meaning
0b0	Bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU60, bit [26]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry.

HWU60	Meaning
0b0	Bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU59, bit [25]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry.

HWU59	Meaning
0b0	Bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD, bit [24]**When FEAT_HPDS is implemented:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBRO_EL2](#).

HPD	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

Note	
In this case, bit[61] (APTable[0]) and bit[59] (PXNTable) of the next level descriptor attributes are required to be ignored by the PE and are no longer reserved, allowing them to be used by software.	

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [23]

Reserved, RES1.

HD, bit [22]**When FEAT_HAFDBS is implemented:**

Hardware management of dirty state in stage 1 translations from EL2.

HD	Meaning
0b0	Stage 1 hardware management of dirty state disabled.
0b1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HA, bit [21]**When FEAT_HAFDBS is implemented:**

Hardware Access flag update in stage 1 translations from EL2.

HA	Meaning
0b0	Stage 1 Access flag update disabled.
0b1	Stage 1 Access flag update enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBI, bit [20]

Top Byte Ignored. Indicates whether the top byte of an address is used for address match for the [TTBR0_EL2](#) region, or ignored and used for tagged addresses.

For more information, see 'Address tagging in AArch64 state'.

TBI	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If FEAT_PAuth is implemented and TCR_EL2.TBID is 1, then this field only applies to Data accesses.

If the value of TBI is 1, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL2.
- An exception taken to EL2.
- An exception return to EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [19]

Reserved, RES0.

PS, bits [18:16]

Physical Address Size.

PS	Meaning	Applies when
0b000	32 bits, 4GB.	
0b001	36 bits, 64GB.	
0b010	40 bits, 1TB.	
0b011	42 bits, 4TB.	
0b100	44 bits, 16TB.	
0b101	48 bits, 256TB.	
0b110	52 bits, 4PB.	
0b111	56 bits, 64PB.	When FEAT_D128 is implemented

All other values are reserved.

The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

If the translation granule is not 64KB and FEAT_LPA2 is not implemented, the value 0b110 is treated as reserved.

It is IMPLEMENTATION DEFINED whether an implementation that does not implement FEAT_LPA supports setting the value of 0b110 for the 64KB translation granule size or whether setting this value behaves as the 0b101 encoding.

~~If in an implementation that supports 52-bit PAs, if the value of this field is not~~ [ID_AA64MMFR0_EL1.PARange](#) is 0b0110, and the value of this field is not 0b110 or a value treated as 0b110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL2 are 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG0, bits [15:14]

Granule size for the [TTBR0_EL2](#).

TG0	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

Other values are reserved.

If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

SH0	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

ORGN0	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

IRGN0	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

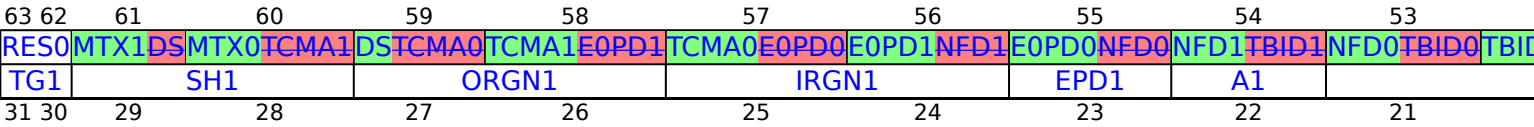
For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_VHE is implemented and HCR_EL2.E2H == 1:



Any of the bits in TCR_EL2 are permitted to be cached in a TLB.

Bits [63:6260]

Reserved, RES0.

MTX1, bit [61]

When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:

Extended memory tag checking.

This field controls address generation and tag checking when EL0 and EL2 are using AArch64 where the data address would be translated by tables pointed to by [TTBRI_EL2](#).

This control has an effect regardless of whether stage 1 of the EL2&0 translation regime is enabled or not.

MTX1	Meaning
0b0	This control has no effect on the PE.
0b1	Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply: <ul style="list-style-type: none">Bits[59:56] are treated as 0b1111 when checking if the address is out of range.If FEAT_PAAuth is implemented, bits[59:56] are not part of the PAC field.A Canonical Tag Check operation is performed on Tag Checked memory accesses to a Canonically Tagged memory location.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MTX0, bit [60]

When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:

Extended memory tag checking.

This field controls address generation and tag checking when EL0 and EL2 are using AArch64 where the data address would be translated by tables pointed to by [TTBRO_EL2](#).

This control has an effect regardless of whether stage 1 of the EL2&0 translation regime is enabled or not.

MTX0	Meaning
0b0	This control has no effect on the PE.
0b1	Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply: <ul style="list-style-type: none"> Bits[59:56] are treated as 0b0000 when checking if the address is out of range. If FEAT_PAAuth is implemented, bits[59:56] are not part of the PAC field. A Canonical Tag Check operation is performed on Tag Checked memory accesses to a Canonically Tagged memory location.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DS, bit [59]

When FEAT_LPA2 is implemented:

This field affects whether a 52-bit output address addressing can when be using described 4KB by and the 16KB translation tables granules in stage 1 of the 4KB EL2&0 or 16KB translation granules regime.

DS	Meaning
0b0	<p>Bits[49:48] of translation descriptors are RES0.</p> <p>Bits[9:8] in Block and Page descriptors encode shareability information in the SH[1:0] field. Bits[9:8] in table descriptors are ignored by hardware.</p> <p>The minimum value of the TCR_EL2.{T0SZ, T1SZ} fields is 16. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>Output address[51:48] is 0b0000.</p>
0b1	<p>Bits[49:48] of translation descriptors hold output address[49:48].</p> <p>Bits[9:8] of Translation table descriptors hold output address[51:50].</p> <p>The shareability information of Block and Page descriptors for cacheable locations is determined by:</p> <ul style="list-style-type: none"> • TCR_EL2.SH0 if the VA is an address that is translated using tables pointed to by TTBR0_EL2. • TCR_EL2.SH1 if the VA is an address that is translated using tables pointed to by TTBR1_EL2. <p>The minimum value of the TCR_EL2.{T0SZ, T1SZ} fields is 12. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>All calculations of the stage 1 base address are modified for tables of fewer than 16 entries so that the table is aligned to 64 bytes.</p> <p>Bits[5:2] of TTBR0_EL2 or TTBR1_EL2 are used to hold bits[51:48] of the output address in all cases.</p> <hr/> <p>Note</p> <p>As FEAT_LVA must be implemented if TCR_EL2.DS == 1, the minimum value of the TCR_EL2.{T0SZ, T1SZ} fields is 12, as determined by that extension.</p> <hr/> <p>For the TLBI Range instructions affecting VA, the format of the argument is changed so that bits[36:0] hold BaseADDR[52:16]. For the 4KB translation granule, bits[15:12] of BaseADDR are treated as 0b0000. For the 16KB translation granule, bits[15:14] of BaseADDR are treated as 0b00.</p> <hr/> <p>Note</p> <p>This forces alignment of the ranges used by the TLBI range instructions.</p> <hr/>

This field is RES0 for a 64KB translation granule.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCMA1, bit [58]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL2, and at EL0 if HCR_EL2.TGE=1, when address[59:55] = 0b11111.

TCMA1	Meaning
0b0	This control has no effect on the generation of Unchecked accesses at EL2 or EL0.
0b1	All accesses are Unchecked.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TCMA0, bit [57]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL2, and at EL0 if HCR_EL2.TGE=1, when address[59:55] = 0b00000.

TCMA0	Meaning
0b0	This control has no effect on the generation of Unchecked accesses at EL2 or EL0.
0b1	All accesses are Unchecked.

Note

Software may change this control bit on a context switch.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EOPD1, bit [56]

When FEAT_EOPD is implemented:

Faulting control for Unprivileged access to any address translated by [TTBR1_EL2](#).

EOPD1	Meaning
0b0	Unprivileged access to any address translated by TTBR1_EL2 will not generate a fault by this mechanism.
0b1	Unprivileged access to any address translated by TTBR1_EL2 will generate a level 0 Translation fault.

Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EOPD0, bit [55]**When FEAT_EOPD is implemented:**

Faulting control for Unprivileged access to any address translated by [TTBR0_EL2](#).

EOPD0	Meaning
0b0	Unprivileged access to any address translated by TTBR0_EL2 will not generate a fault by this mechanism.
0b1	Unprivileged access to any address translated by TTBR0_EL2 will generate a level 0 Translation fault.

Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NFD1, bit [54]**When FEAT_SVE is implemented or FEAT_TME is implemented:**

Non-fault translation table walk disable for stage 1 translations using [TTBR1_EL2](#).

This bit controls whether to perform a stage 1 translation table walk in response to a non-fault unprivileged access for a virtual address that is translated using [TTBR1_EL2](#).

If SVE is implemented, the affected access types include:

- All accesses due to an SVE non-fault contiguous load instruction.
- Accesses due to an SVE first-fault gather load instruction that are not for the First active element. Accesses due to an SVE first-fault contiguous load instruction are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

For more information, see 'The Scalable Vector Extension (SVE)'.

If FEAT_TME is implemented, the affected access types include all accesses generated by a load or store instruction in Transactional state.

NFD1	Meaning
0b0	Does not disable stage 1 translation table walks using TTBR1_EL2 .
0b1	A TLB miss on a virtual address that is translated using TTBR1_EL2 due to the specified access types causes the access to fail without taking an exception. No stage 1 translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NFD0, bit [53]**When FEAT_SVE is implemented or FEAT_TME is implemented:**

Non-fault translation table walk disable for stage 1 translations using [TTBR0_EL2](#).

This bit controls whether to perform a stage 1 translation table walk in response to a non-fault unprivileged access for a virtual address that is translated using [TTBR0_EL2](#).

If SVE is implemented, the affected access types include:

- All accesses due to an SVE non-fault contiguous load instruction.
- Accesses due to an SVE first-fault gather load instruction that are not for the First active element. Accesses due to an SVE first-fault contiguous load instruction are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

For more information, see 'The Scalable Vector Extension (SVE)'.

If FEAT_TME is implemented, the affected access types include all accesses generated by a load or store instruction in Transactional state.

NFD0	Meaning
0b0	Does not disable stage 1 translation table walks using TTBR0_EL2 .
0b1	A TLB miss on a virtual address that is translated using TTBR0_EL2 due to the specified access types causes the access to fail without taking an exception. No stage 1 translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID1, bit [52]**When FEAT_PAuth is implemented:**

Controls the use of the top byte of instruction addresses for address matching.

For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.

For more information, see 'Address tagging in AArch64 state'.

TBID1	Meaning
0b0	TCR_EL2.TBI1 applies to Instruction and Data accesses.
0b1	TCR_EL2.TBI1 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR1_EL2](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID0, bit [51]**When FEAT_PAAuth is implemented:**

Controls the use of the top byte of instruction addresses for address matching.

For more information, see 'Address tagging in AArch64 state'.

TBID0	Meaning
0b0	TCR_EL2.TBI0 applies to Instruction and Data accesses.
0b1	TCR_EL2.TBI0 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL2](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU162, bit [50]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

HWU162	Meaning
0b0	For translations using TTBR1_EL2 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL2 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU161, bit [49]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

HWU161	Meaning
0b0	For translations using TTBR1_EL2 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL2 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU160, bit [48]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

HWU160	Meaning
0b0	For translations using TTBR1_EL2 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL2 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU159, bit [47]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

HWU159	Meaning
0b0	For translations using TTBR1_EL2 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR1_EL2 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU062, bit [46]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU062	Meaning
0b0	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU061, bit [45]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU061	Meaning
0b0	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU060, bit [44]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU060	Meaning
0b0	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU059, bit [43]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

HWU059	Meaning
0b0	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD1, bit [42]

When FEAT_HPDS is implemented:

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR1_EL2](#).

HPD1	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD0, bit [41]**When FEAT_HPDS is implemented:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR0_EL2](#).

HPD0	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HD, bit [40]**When FEAT_HAFDBS is implemented:**

Hardware management of dirty state in stage 1 translations from EL2.

HD	Meaning
0b0	Stage 1 hardware management of dirty state disabled.
0b1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HA, bit [39]**When FEAT_HAFDBS is implemented:**

Hardware Access flag update in stage 1 translations from EL2.

HA	Meaning
0b0	Stage 1 Access flag update disabled.
0b1	Stage 1 Access flag update enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBI1, bit [38]

Top Byte Ignored. Indicates whether the top byte of an address is used for address match for the [TTBR1_EL2](#) region, or ignored and used for tagged addresses.

For more information, see 'Address tagging in AArch64 state'.

TBI1	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR1_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If FEAT_PAuth is implemented and TCR_EL2.TBID1 is 1, then this field only applies to Data accesses.

If the value of TBI1 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TBI0, bit [37]

Top Byte Ignored. Indicates whether the top byte of an address is used for address match for the [TTBR0_EL2](#) region, or ignored and used for tagged addresses.

For more information, see 'Address tagging in AArch64 state'.

TBI0	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If FEAT_PAuth is implemented and TCR_EL2.TBID0 is 1, then this field only applies to Data accesses.

If the value of TBI0 is 1 and bit [55] of the target address to be stored to the PC is 0, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

AS, bit [36]

ASID Size.

AS	Meaning
0b0	8 bit - the upper 8 bits of TTBR0_EL2 and TTBR1_EL2 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros for when used for allocation and matching entries in the TLB.
0b1	16 bit - the upper 16 bits of TTBR0_EL2 and TTBR1_EL2 are used for allocation and matching in the TLB.

If the implementation has only 8 bits of ASID, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [35]

Reserved, RES0.

IPS, bits [34:32]

Intermediate Physical Address Size.

IPS	Meaning	Applies when
0b000	32 bits, 4GB.	
0b001	36 bits, 64GB.	
0b010	40 bits, 1TB.	
0b011	42 bits, 4TB.	
0b100	44 bits, 16TB.	
0b101	48 bits, 256TB.	
0b110	52 bits, 4PB.	When FEAT_LPA is implemented

All other values are reserved.

The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

If the translation granule is not 64KB, the value 0b110 is treated as reserved.

It is IMPLEMENTATION DEFINED whether an implementation that does not implement FEAT_LPA supports setting the value of 0b110 for the 64KB translation granule size or whether setting this value behaves as the 0b101 encoding.

If in an implementation that supports 52-bit PAs, if the value of this field is not [ID_AA64MMFR0_EL1.PARange](#) is 0b0110, and the value of this field is not 0b110 or a value treated as 0b110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL2 are 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG1, bits [31:30]

Granule size for the [TTBR1_EL2](#).

TG1	Meaning
0b01	16KB.
0b10	4KB.
0b11	64KB.

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH1, bits [29:28]

Shareability attribute for memory associated with translation table walks using [TTBR1_EL2](#).

SH1	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN1, bits [27:26]

Outer cacheability attribute for memory associated with translation table walks using [TTBR1_EL2](#).

ORGN1	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN1, bits [25:24]

Inner cacheability attribute for memory associated with translation table walks using [TTBR1_EL2](#).

IRGN1	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EPD1, bit [23]

Translation table walk disable for translations using [TTBR1_EL2](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1_EL2](#). The encoding of this bit is:

EPD1	Meaning
0b0	Perform translation table walks using TTBR1_EL2 .
0b1	A TLB miss on an address that is translated using TTBR1_EL2 generates a Translation fault. No translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

A1, bit [22]

Selects whether [TTBR0_EL2](#) or [TTBR1_EL2](#) defines the ASID. The encoding of this bit is:

A1	Meaning
0b0	TTBR0_EL2 .ASID defines the ASID.
0b1	TTBR1_EL2 .ASID defines the ASID.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T1SZ, bits [21:16]

The size offset of the memory region addressed by [TTBR1_EL2](#). The region size is $2^{(64-T1SZ)}$ bytes.

The maximum and minimum possible values for T1SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG0, bits [15:14]

Granule size for the [TTBR0_EL2](#).

TG0	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

SH0	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGNO, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

ORGNO	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGNO, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

IRGNO	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

EPD0, bit [7]

Translation table walk disable for translations using [TTBR0_EL2](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR0_EL2](#). The encoding of this bit is:

EPD0	Meaning
0b0	Perform translation table walks using TTBR0_EL2 .
0b1	A TLB miss on an address that is translated using TTBR0_EL2 generates a Translation fault. No translation table walk is performed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for TOSZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TCR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TCR_EL2 or TCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR_EL2;

```

MSR TCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    TCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TCR_EL2 = X[t, 64];

```

MRS <Xt>, TCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x120];
    else
        X[t, 64] = TCR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TCR_EL2;
    else
        X[t, 64] = TCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR_EL1;

```

MSR TCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.TCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x120] = X[t, 64];
    else
        TCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TCR_EL2 = X[t, 64];
    else
        TCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TCR EL3 characteristics are:

Purpose

The control register for stage 1 of the EL3 translation regime.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to TCR_EL3 are UNDEFINED.

Attributes

TCR EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39					
RES0																				DisCH0DS		HAFT		PTTWI		RES0		D	
RES1	TCMAT	TBID	HWU62	HWU61	HWU60	HWU59	HPD	RES1	HD	HA	TBI	RES0	PS	TG0	SH0	ORGNO			IRGNO			RE							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7					

Unless stated otherwise, any of the bits in TCR EL3 are permitted to be cached in a TLB.

Bits [63:4433]

Reserved, RES0.

DisCH0, bit [43]

When FEAT D128 is implemented and TCR EL3.D128 == 1:

Disable Contiguous Hint for Start Table.

DisCH0	Meaning
0b0	Contiguous Hint of Block or Page descriptors of the Start Table are not affected by this field.
0b1	Contiguous Hint of Block or Page descriptors of the Start Table are disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HAFT, bit [42]

When FEAT HAFT is implemented:

Hardware managed Access Flag for Tables.

Enables the Hardware managed Access Flag for Tables.

HAFT	Meaning
0b0	Hardware managed Access Flag for Tables is disabled.
0b1	Hardware managed Access Flag for Tables is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PTTWI, bit [41]

When FEAT_TTE is implemented:

Permit Translation table walk Incoherence.

Permits RCWS instructions to have Reduced Coherence property.

PTTWI	Meaning
0b0	Write accesses generated by RCWS at EL3 do not have the Reduced Coherence property.
0b1	Write accesses generated by RCWS at EL3 have the Reduced Coherence property.

This bit is permitted to be built as a read-only bit with a fixed value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [40:39]

Reserved, RES0.

D128, bit [38]

When FEAT_D128 is implemented:

Enable 128-bit Page Table Descriptors.

Enables VMSAv9-128 translation system for the Stage 1 EL3 Translation Process.

D128	Meaning
0b0	Translation system follows VMSA-64 translation process.
0b1	Translation system follows VMSAv9-128 translation process.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AIE, bit [37]**When FEAT_AIE is implemented:**

Enable Attribute Indexing Extension. Control for Attribute Indexing Extension for Stage 1 EL3 Translation Process.

AIE	Meaning
0b0	Attribute Indexing Extension Disabled.
0b1	Attribute Indexing Extension Enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

POE, bit [36]**When FEAT_S1POE is implemented:**

POE. Controls setting of permission overlay for EL3 accesses in stage 1 of the EL3 translation regime.

POE	Meaning
0b0	Permission overlay disabled for EL3 access in stage 1 of EL3 translation regime..
0b1	Permission overlay enabled for EL3 access in stage 1 of EL3 translation regime.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PIE, bit [35]**When FEAT_S1PIE is implemented:**

Select Permission Model. Controls setting of indirect permission model in Stage 1 EL3 Translation Process.

PIE	Meaning
0b0	Direct permission model.
0b1	Indirect permission model.

This field is RES1 when TCR_EL3.D128 is set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PnCH, bit [34]**When FEAT_TTE is implemented:**

Protected attribute enable. Indicates use of bit[52] of the stage 1 translation table entry for translations using [TTBR0_EL3](#).

PnCH	Meaning
0b0	For translations using TTBR0_EL3 , bit[52] of each stage 1 translation table entry does not indicate protected attribute.
0b1	For translations using TTBR0_EL3 , bit[52] of each stage 1 translation table entry indicates protected attribute.

This field is RES1 when TCR_EL3.D128 is set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MTX, bit [33]**When FEAT_MTE_NO_ADDRESS_TAGS is implemented or FEAT_MTE_CANONICAL_TAGS is implemented:**

Extended memory tag checking.

This field controls address generation and tag checking when EL3 is using AArch64 where the data address would be translated by tables pointed to by [TTBR0_EL3](#).

This control has an effect regardless of whether stage 1 of the EL3 translation regime is enabled or not.

MTX	Meaning
0b0	This control has no effect on the PE.
0b1	Bits[59:56] of a 64-bit VA hold a Logical Address Tag, and all of the following apply: <ul style="list-style-type: none"> Bits[59:56] are treated as 0b0000 when checking if the address is out of range. If FEAT_PAuth is implemented, bits[59:56] are not part of the PAC field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

DS, bit [32]**When FEAT_LPA2 is implemented:**

This field affects whether a 52-bit output address addressing can when be using described 4KB by and the 16KB translation tables granules in stage 1 of the 4KB EL3 or 16KB translation granules regime.

DS	Meaning
0b0	<p>Bits[49:48] of translation descriptors are RES0.</p> <p>Bits[9:8] in Block and Page descriptors encode shareability information in the SH[1:0] field. Bits[9:8] in Table descriptors are ignored by hardware.</p> <p>The minimum value of TCR_EL3.T0SZ is 16. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>Output address[51:48] is 0b0000.</p>
0b1	<p>Bits[49:48] of translation descriptors hold output address[49:48].</p> <p>Bits[9:8] of table translation descriptors hold output address[51:50].</p> <p>The shareability information of Block and Page descriptors for cacheable locations is determined by TCR_EL3.SH0.</p> <p>The minimum value of TCR_EL3.T0SZ is 12. Any memory access using a smaller value generates a stage 1 level 0 translation table fault.</p> <p>All calculations of the stage 1 base address are modified for tables of fewer than 8 entries so that the table is aligned to 64 bytes.</p> <p>Bits[5:2] of TTBR0_EL3 are used to hold bits[51:48] of the output address in all cases.</p> <hr/> <p>Note</p> <p>As FEAT_LVA must be implemented if TCR_EL3.DS == 1, the minimum value of the TCR_EL3.T0SZ field is 12, as determined by that extension.</p> <hr/> <p>For the TLBI Range instructions affecting VA, the format of the argument is changed so that bits[36:0] hold BaseADDR[52:16]. For the 4KB translation granule, bits[15:12] of BaseADDR are treated as 0b0000. For the 16KB translation granule, bits[15:14] of BaseADDR are treated as 0b00.</p> <hr/> <p>Note</p> <p>This forces alignment of the ranges used by the TLBI range instructions.</p> <hr/>

This field is RES0 for a 64KB translation granule.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [31]

Reserved, RES1.

TCMA, bit [30]

When FEAT_MTE2 is implemented:

Controls the generation of Unchecked accesses at EL3 when address [59:56] = 0b0000.

TCMA	Meaning
0b0	This control has no effect on the generation of Unchecked accesses.
0b1	All accesses are Unchecked.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBID, bit [29]**When FEAT_PAAuth is implemented:**

Controls the use of the top byte of instruction addresses for address matching.

TBID	Meaning
0b0	TCR_EL3.TBI applies to Instruction and Data accesses.
0b1	TCR_EL3.TBI applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL3](#).

For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU62, bit [28]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry.

HWU62	Meaning
0b0	Bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL3.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL3.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU61, bit [27]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry.

HWU61	Meaning
0b0	Bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL3.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL3.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU60, bit [26]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry.

HWU60	Meaning
0b0	Bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL3.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL3.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU59, bit [25]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry.

HWU59	Meaning
0b0	Bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL3.HPD is 1.

The Effective value of this field is 0 if the value of TCR_EL3.HPD is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HPD, bit [24]**When FEAT_HPDS is implemented:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBRO_EL3](#).

HPD	Meaning
0b0	Hierarchical permissions are enabled.
0b1	Hierarchical permissions are disabled.

Note
In this case, bit[61] (APTable[0]) and bit[59] (PXNTable) of the next level descriptor attributes are required to be ignored by the PE, and are no longer reserved, allowing them to be used by software.

When disabled, the permissions are treated as if the bits are zero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [23]

Reserved, RES1.

HD, bit [22]**When FEAT_HAFDBS is implemented:**

Hardware management of dirty state in stage 1 translations from EL3.

HD	Meaning
0b0	Stage 1 hardware management of dirty state disabled.
0b1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HA, bit [21]**When FEAT_HAFDBS is implemented:**

Hardware Access flag update in stage 1 translations from EL3.

HA	Meaning
0b0	Stage 1 Access flag update disabled.
0b1	Stage 1 Access flag update enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TBI, bit [20]

Top Byte Ignored. Indicates whether the top byte of an address is used for address match for the [TTBR0_EL3](#) region, or ignored and used for tagged addresses.

TBI	Meaning
0b0	Top Byte used in the address calculation.
0b1	Top Byte ignored in the address calculation.

This affects addresses generated in EL3 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL3](#). It has an effect whether the EL3 translation regime is enabled or not.

If FEAT_PAAuth is implemented and TCR_EL3.TBID is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI is 1, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL3.
- A exception taken to EL3.
- An exception return to EL3.

For more information, see 'Address tagging in AArch64 state'.

Note

This control determines the scope of address tagging. It never causes an exception to be generated.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [19]

Reserved, RES0.

PS, bits [18:16]

Physical Address Size.

PS	Meaning	Applies when
0b000	32 bits, 4GB.	
0b001	36 bits, 64GB.	
0b010	40 bits, 1TB.	
0b011	42 bits, 4TB.	
0b100	44 bits, 16TB.	
0b101	48 bits, 256TB.	
0b110	52 bits, 4PB.	
0b111	56 bits, 64PB.	When FEAT_D128 is implemented

All other values are reserved.

The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

If the translation granule is not 64KB and FEAT_LPA2 is not implemented, the value 0b110 is treated as reserved.

It is IMPLEMENTATION DEFINED whether an implementation that does not implement FEAT_LPA supports setting the value of 0b110 for the 64KB translation granule size or whether setting this value behaves as the 0b101 encoding.

If in an implementation that supports 52-bit PAs, if the value of this field is not ID_AA64MMFR0_EL1.PARange is 0b0110, and the value of this field is not 0b110 or a value treated as 0b110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL3 are 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG0, bits [15:14]

Granule size for the [TTBR0_EL3](#).

TG0	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

Other values are reserved.

If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL3](#).

SH0	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL3](#).

ORGN0	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL3](#).

IRGN0	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [7:6]

Reserved, RES0.

TOSZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL3](#). The region size is $2^{(64-TOSZ)}$ bytes.

The maximum and minimum possible values for TOSZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TCR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TCR_EL3;

```


MSR_TCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    TCR EL3 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI ASIDE1, TLBI ASIDE1NXS, TLB Invalidate by ASID, EL1

The TLBI ASIDE1, TLBI ASIDE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate an address using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate an address using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI ASIDE1, TLBI ASIDE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Bits [47:0]

Reserved, RES0.

Executing the TLBI ASIDE1, TLBI ASIDE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI ASIDE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIASIDE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBI_AllAttr, X[t, 64]);

```

TLBI ASIDE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0111	0b010
------	-------	--------	--------	-------

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIASIDE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI ASIDE1IS, TLBI ASIDE1ISNXS, TLB Invalidate by ASID, EL1, Inner Shareable

The TLBI ASIDE1IS, TLBI ASIDE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate an address using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate an address using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI ASIDE1IS, TLBI ASIDE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Bits [47:0]

Reserved, RES0.

Executing the TLBI ASIDE1IS, TLBI ASIDE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI ASIDE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIASIDE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBI_AllAttr, X[t, 64]);

```

TLBI ASIDE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b010

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIASIDE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI ASIDE1OS, TLBI ASIDE1OSNXS, TLB Invalidate by ASID, EL1, Outer Shareable

The TLBI ASIDE1OS, TLBI ASIDE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate an address using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate an address using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI ASIDE1OS, TLBI ASIDE1OSNXS are UNDEFINED.

Attributes

TLBI ASIDE1OS, TLBI ASIDE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																RES0															
																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Bits [47:0]

Reserved, RES0.

Executing the TLBI ASIDE1OS, TLBI ASIDE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI ASIDE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIASIDE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBI_AllAttr, X[t, 64]);

```

TLBI ASIDE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b010

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIASIDE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_ASID(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI IPAS2E1, TLBI IPAS2E1NXS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1

The TLBI IPAS2E1, TLBI IPAS2E1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - SCR_EL3.{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - SCR_EL3.NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - SCR_EL3.NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI IPAS2E1, TLBI IPAS2E1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
NS	RES0														TTL			RES0			IPA[51:48]				IPA[47:12]									
IPA[47:12]																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

NS, bit [63]

When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]

When FEAT_LPA is implemented:

Extension to IPA[47:12]. For more information, see IPA[47:12].

Otherwise:

Reserved, RES0.

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2E1, TLBI IPAS2E1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI IPAS2E1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI IPAS2E1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

The TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

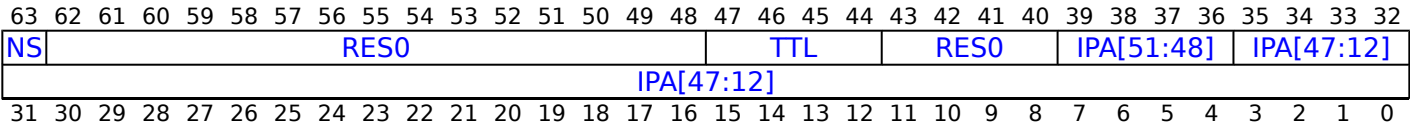
Configuration

There are no configuration notes.

Attributes

TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS is a 64-bit System instruction.

Field descriptions



NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]

When FEAT_LPA is implemented:

Extension to IPA[47:12]. For more information, see IPA[47:12].

Otherwise:

Reserved, RES0.

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI IPAS2E1ISNXS{, <Xt>}

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

The TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

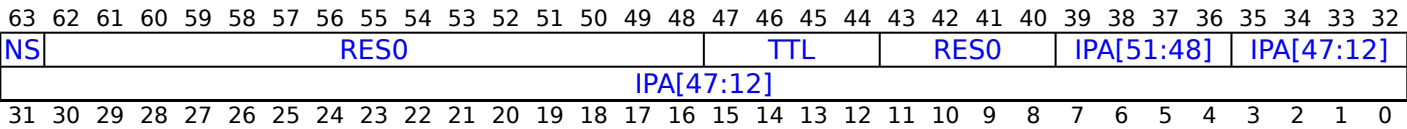
Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS are UNDEFINED.

Attributes

TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS is a 64-bit System instruction.

Field descriptions



NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44] When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]

Extension to IPA[47:12]. For more information, see IPA[47:12].

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI IPAS2E1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b000

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b000

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

TLBI IPAS2LE1, TLBI IPAS2LE1NXS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

The TLBI IPAS2LE1, TLBI IPAS2LE1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

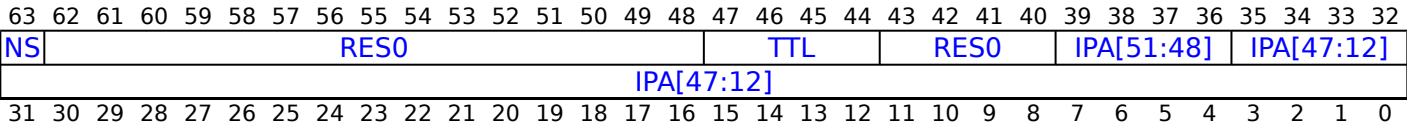
Configuration

There are no configuration notes.

Attributes

TLBI IPAS2LE1, TLBI IPAS2LE1NXS is a 64-bit System instruction.

Field descriptions



NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]

When FEAT_LPA is implemented:

Extension to IPA[47:12]. For more information, see IPA[47:12].

Otherwise:

Reserved, RES0.

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2LE1, TLBI IPAS2LE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI IPAS2LE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI IPAS2LE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

The TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

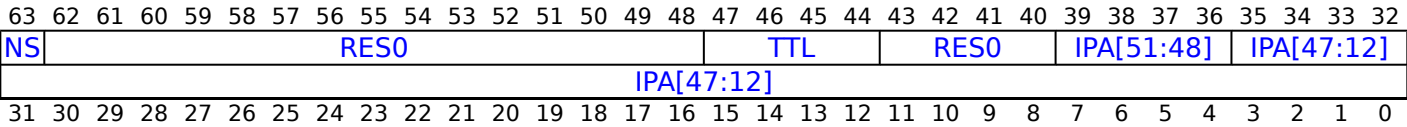
Configuration

There are no configuration notes.

Attributes

TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS is a 64-bit System instruction.

Field descriptions



NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]
When FEAT_LPA is implemented:

Extension to IPA[47:12]. For more information, see IPA[47:12].

Otherwise:

Reserved, RES0.

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI IPAS2LE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0000	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
    end
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    end
end

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

The TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

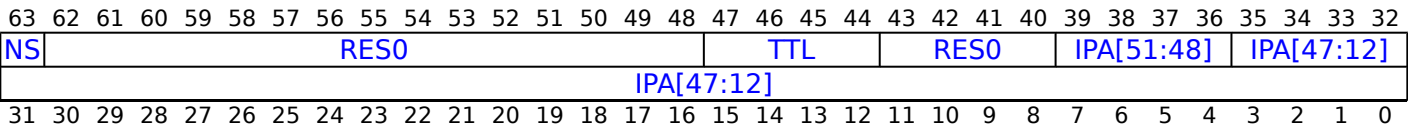
Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS are UNDEFINED.

Attributes

TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS is a 64-bit System instruction.

Field descriptions



NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:40]

Reserved, RES0.

IPA[51:48], bits [39:36]

Extension to IPA[47:12]. For more information, see IPA[47:12].

IPA[47:12], bits [35:0]

Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0.

If ID_AA64MMFR0_EL1.PARange is 0b0110, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

When FEAT_LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

Executing the TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI IPAS2LE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
```

TLBI IPAS2LE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b100

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)

htmldiff from-

(new)

TLBI RIPAS2E1, TLBI RIPAS2E1NXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1

The TLBI RIPAS2E1, TLBI RIPAS2E1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. ~~walk.~~

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.

- If FEAT_RME is implemented, one of the following applies:
 - SCR_EL3.{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - SCR_EL3.NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the ~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseAddr[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseAddr[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseAddr[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseAddr[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseAddr[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RIPAS2E1, TLBI RIPAS2E1NXS are UNDEFINED.

Attributes

TLBI RIPAS2E1, TLBI RIPAS2E1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR									
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

TLBI RIPAS2E1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RIPAS2E1NXS{, <Xt>}

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
    endif
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
    endif
endif
end

```

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

The TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. walk.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.

- If FEAT_RME is implemented, one of the following applies:
 - SCR_EL3.{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - SCR_EL3.NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseAddr[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseAddr[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseAddr[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseAddr[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseAddr[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS are UNDEFINED.

Attributes

TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR									
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

(old)

htmldiff from-

(new)

TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

The TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. ~~walk.~~

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.

- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the ~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS are UNDEFINED.

Attributes

TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR									
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate level 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate level 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate level 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RIPAS2E1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b011

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
```

TLBI RIPAS2E1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b011

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

The TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the leaf~~final~~ level of the translation table walk, indicated by the TTL hint~~walk~~.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this System instruction.

For 64-bit translation table entry, the~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS are UNDEFINED.

Attributes

TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR									
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RIPAS2LE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

```
TLBI RIPAS2LE1NXS{, <Xt>}
```

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b110

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

The TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the leaf~~final~~ level of the translation table walk, indicated by the TTL hint~~walk~~.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS are UNDEFINED.

Attributes

TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR									
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

```
TLBI RIPAS2LE1IS{, <Xt>}
```

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

```
TLBI RIPAS2LE1ISNXS{, <Xt>}
```

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

The TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 2 only translation table entry, from the leaf~~final~~ level of the translation table walk, indicated by the TTL hint walk.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - SCR_EL3.{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - SCR_EL3.{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - SCR_EL3.NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - SCR_EL3.NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
- A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS are UNDEFINED.

Attributes

TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
NS	RES0														TG	SCALE	NUM					TTL	BaseADDR										
BaseADDR																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate level 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate level 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate level 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RIPAS2LE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
```

TLBI RIPAS2LE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAAE1, TLBI RVAAE1NXS, TLB Range Invalidate by VA, All ASID, EL1

The TLBI RVAAE1, TLBI RVAAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. walk.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAAE1, TLBI RVAAE1NXS are UNDEFINED.

Attributes

TLBI RVAAE1, TLBI RVAAE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																TG	SCALE	NUM				TTL		BaseADDR								
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAAE1, TLBI RVAAE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIRVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
                elsif PSTATE.EL == EL3 then
                    if HCR_EL2.<E2H,TGE> == '11' then
                        AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
                    else
                        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAAE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAAE1IS, TLBI RVAAE1ISNXS, TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

The TLBI RVAAE1IS, TLBI RVAAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. walk.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:

- If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
- If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAAE1IS, TLBI RVAAE1ISNXS are UNDEFINED.

Attributes

TLBI RVAAE1IS, TLBI RVAAE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG		SCALE		NUM					TTL		BaseADDR				
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAAE1IS, TLBI RVAAE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAAE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAAE1OS, TLBI RVAAE1OSNXS, TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

The TLBI RVAAE1OS, TLBI RVAAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. walk.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:

TLBI RVAAE1OS, TLBI RVAAE1OSNXS, TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

- If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
 - For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVAAE1OS, TLBI RVAAE1OSNXS are UNDEFINED.

Attributes

TLBI RVAAE1OS, TLBI RVAAE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG	SCALE	NUM					TTL		BaseADDR						
																BaseADDR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAAE1OS, TLBI RVAAE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAAE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAALE1, TLBI RVAALE1NXS, TLB Range Invalidate by VA, All ASID, Last level, EL1

The TLBI RVAALE1, TLBI RVAALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the leaf~~final~~ level of the translation table walk, indicated by the TTL hint.~~walk~~.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAALE1, TLBI RVAALE1NXS are UNDEFINED.

Attributes

TLBI RVAALE1, TLBI RVAALE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG	SCALE	NUM				TTL		BaseADDR							
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAALE1, TLBI RVAALE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIRVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAALE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)	htmldiff from-	(new)
-------	----------------	-------

TLBI RVAALE1IS, TLBI RVAALE1ISNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

The TLBI RVAALE1IS, TLBI RVAALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the leaf~~final~~ level of the translation table ~~walk~~, indicated by the TTL hint.~~walk~~.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
- A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAALE1IS, TLBI RVAALE1ISNXS are UNDEFINED.

Attributes

TLBI RVAALE1IS, TLBI RVAALE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG	SCALE	NUM				TTL		BaseADDR							
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate level 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate level 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate level 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAALE1IS, TLBI RVAALE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAALE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAALE1OS, TLBI RVAALE1OSNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

The TLBI RVAALE1OS, TLBI RVAALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the leaf~~final~~ level of the translation table walk, indicated by the TTL hint.~~walk~~.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
- A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVAALE1OS, TLBI RVAALE1OSNXS are UNDEFINED.

Attributes

TLBI RVAALE1OS, TLBI RVAALE1OSNXS is a 64-bit System instruction.

Field descriptions

62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																TG	SCALE	NUM				TTL		BaseADDR							
BaseADDR																															
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate level 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate level 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate level 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAALE1OS, TLBI RVAALE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAALE10SNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b111


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE1, TLBI RVAE1NXS, TLB Range Invalidate by VA, EL1

The TLBI RVAE1, TLBI RVAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. ~~entry.~~
- Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the ~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

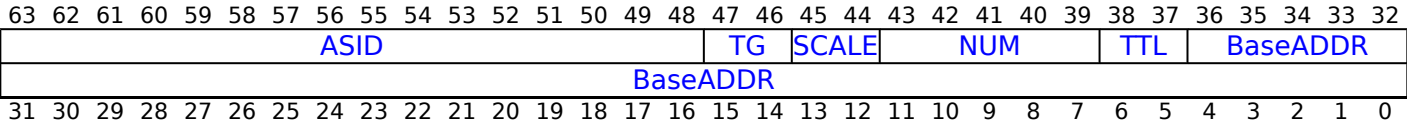
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE1, TLBI RVAE1NXS are UNDEFINED.

Attributes

TLBI RVAE1, TLBI RVAE1NXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE1, TLBI RVAE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE1IS, TLBI RVAE1ISNXS, TLB Range Invalidate by VA, EL1, Inner Shareable

The TLBI RVAE1IS, TLBI RVAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

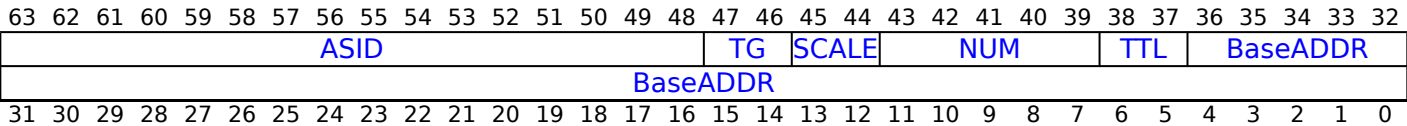
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE1IS, TLBI RVAE1ISNXS are UNDEFINED.

Attributes

TLBI RVAE1IS, TLBI RVAE1ISNXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

- TTL Level hint. The TTL hint is only guaranteed to invalidate: entries in the range that match the level described by the TTL hint.
- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
 - Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE1IS, TLBI RVAE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE1OS, TLBI RVAE1OSNXS, TLB Range Invalidate by VA, EL1, Outer Shareable

The TLBI RVAE1OS, TLBI RVAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 000000000000000.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVAE1OS, TLBI RVAE1OSNXS are UNDEFINED.

Attributes

TLBI RVAE1OS, TLBI RVAE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																TG	SCALE	NUM				TTL		BaseADDR									
BaseADDR																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE1OS, TLBI RVAE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE10SNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVAE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE2, TLBI RVAE2NXS, TLB Range Invalidate by VA, EL2

The TLBI RVAE2, TLBI RVAE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from any level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

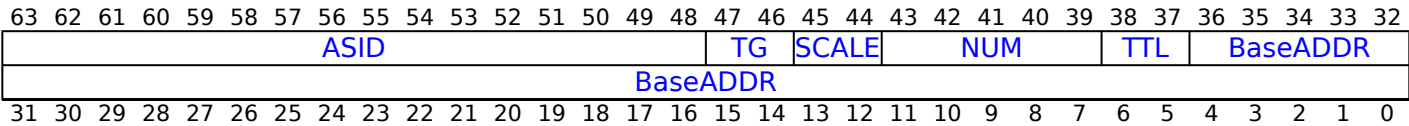
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE2, TLBI RVAE2NXS are UNDEFINED.

Attributes

TLBI RVAE2, TLBI RVAE2NXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]
When HCR_EL2.E2H == 1:

- ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.
- Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.
- If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE2, TLBI RVAE2NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE2NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE2IS, TLBI RVAE2ISNXS, TLB Range Invalidate by VA, EL2, Inner Shareable

The TLBI RVAE2IS, TLBI RVAE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from any level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE2IS, TLBI RVAE2ISNXXS are UNDEFINED.

Attributes

TLBI RVAE2IS, TLBI RVAE2ISNXXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TG	SCALE	NUM				TTL		BaseADDR							
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]
When HCR_EL2.E2H == 1:

- ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.
- Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.
- If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE2IS, TLBI RVAE2ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE2ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0010	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE2OS, TLBI RVAE2OSNXS, TLB Range Invalidate by VA, EL2, Outer Shareable

The TLBI RVAE2OS, TLBI RVAE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint entry.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from any level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVAE2OS, TLBI RVAE2OSNXS are UNDEFINED.

Attributes

TLBI RVAE2OS, TLBI RVAE2OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
ASID																TG	SCALE	NUM					TTL		BaseADDR							
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE2OS, TLBI RVAE2OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE2OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0101	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVAE3, TLBI RVAE3NXS, TLB Range Invalidate by VA, EL3

The TLBI RVAE3, TLBI RVAE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~walk.~~
- Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

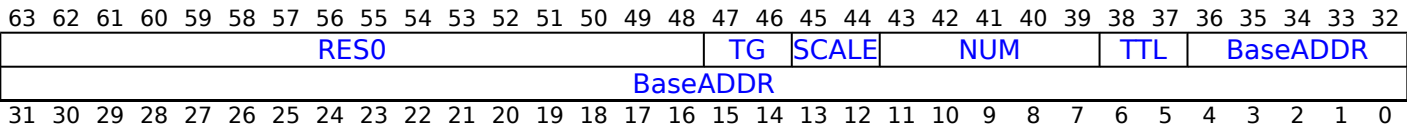
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE3, TLBI RVAE3NXS are UNDEFINED.

Attributes

TLBI RVAE3, TLBI RVAE3NXS is a 64-bit System instruction.

Field descriptions



Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: entries in the range that match the level described by the TTL hint.

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVAE3, TLBI RVAE3NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBIlevel_Any,
    TLBI_AllAttr, X[t, 64]);

```

TLBI RVAE3NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBILevel_Any,
    TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RVAE3IS, TLBI RVAE3ISNXS, TLB Range Invalidate by VA, EL3, Inner Shareable

The TLBI RVAE3IS, TLBI RVAE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~walk.~~
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseAddr[29:12]$ is not equal to 00000000000000000000.
 - If $TTL == 10$ and $BaseAddr[20:12]$ is not equal to 000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseAddr[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseAddr[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseAddr[28:16]$ is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

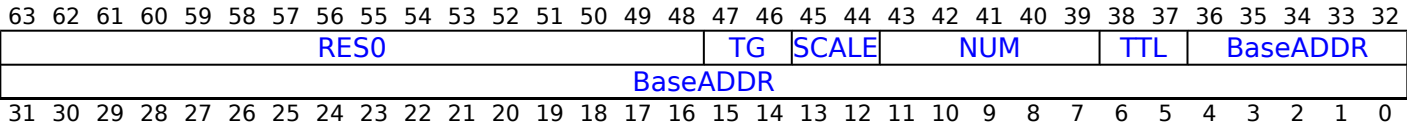
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAE3IS, TLBI RVAE3ISNXS are UNDEFINED.

Attributes

TLBI RVAE3IS, TLBI RVAE3ISNXS is a 64-bit System instruction.

Field descriptions



Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: entries in the range that match the level described by the TTL hint.

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVAE3IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Any,
    TLBI AllAttr, X[t, 64]);

```

TLBI RVAE3ISNXS{, <Xt>}

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Any,
    TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RVAE3OS, TLBI RVAE3OSNXS, TLB Range Invalidate by VA, EL3, Outer Shareable

The TLBI RVAE3OS, TLBI RVAE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~walk.~~
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseAddr[29:12]$ is not equal to 00000000000000000000.
 - If $TTL == 10$ and $BaseAddr[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseAddr[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseAddr[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseAddr[28:16]$ is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVAE3OS, TLBI RVAE3OSNXS are UNDEFINED.

Attributes

TLBI RVAE3OS, TLBI RVAE3OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG	SCALE	NUM					TTL		BaseADDR						
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

The starting address for the range of the maintenance instruction.

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Accesses to this instruction use the following encodings in the System instruction encoding space:

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0101	0b001

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0101	0b001

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TLBI RVALE1, TLBI RVALE1NXS, TLB Range Invalidate by VA, Last level, EL1

The TLBI RVALE1, TLBI RVALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint. ~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the ~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

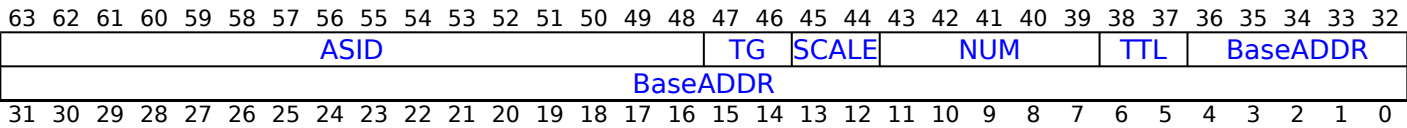
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE1, TLBI RVALE1NXS are UNDEFINED.

Attributes

TLBI RVALE1, TLBI RVALE1NXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**

- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE1, TLBI RVALE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b101


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE1IS, TLBI RVALE1ISNXS, TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

The TLBI RVALE1IS, TLBI RVALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:

TLBI RVALE1IS, TLBI RVALE1ISNXS, TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

- If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
 - For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE1IS, TLBI RVALE1ISNXS are UNDEFINED.

Attributes

TLBI RVALE1IS, TLBI RVALE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TG	SCALE	NUM				TTL		BaseADDR							
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE1IS, TLBI RVALE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE1OS, TLBI RVALE1OSNXS, TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

The TLBI RVALE1OS, TLBI RVALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 64-bit translation table entry, the~~The~~ range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:

TLBI RVALE1OS, TLBI RVALE1OSNXS, TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

- If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
 - For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVALE1OS, TLBI RVALE1OSNXS are UNDEFINED.

Attributes

TLBI RVALE1OS, TLBI RVALE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TG	SCALE	NUM				TTL		BaseADDR							
																BaseADDR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE1OS, TLBI RVALE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE10SNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE2, TLBI RVALE2NXS, TLB Range Invalidate by VA, Last level, EL2

The TLBI RVALE2, TLBI RVALE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[\text{BaseADDR} \leq \text{VA} < \text{BaseADDR} + ((\text{NUM} + 1) * 2^{(5 * \text{SCALE} + 1)} * \text{Translation_Granule_Size})]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

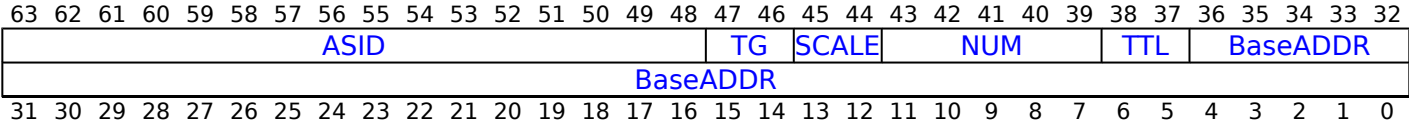
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE2, TLBI RVALE2NXS are UNDEFINED.

Attributes

TLBI RVALE2, TLBI RVALE2NXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]
When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]**When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:**

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE2, TLBI RVALE2NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE2NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE2IS, TLBI RVALE2ISNXS, TLB Range Invalidate by VA, Last level, EL2, Inner Shareable

The TLBI RVALE2IS, TLBI RVALE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint entry.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE2IS, TLBI RVALE2ISNXS are UNDEFINED.

Attributes

TLBI RVALE2IS, TLBI RVALE2ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
ASID																TG	SCALE	NUM					TTL		BaseADDR							
BaseADDR																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- **Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- **Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE2IS, TLBI RVALE2ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE2ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE2OS, TLBI RVALE2OSNXS, TLB Range Invalidate by VA, Last level, EL2, Outer Shareable

The TLBI RVALE2OS, TLBI RVALE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.~~entry.~~
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1) * Translation_Granule_Size})]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

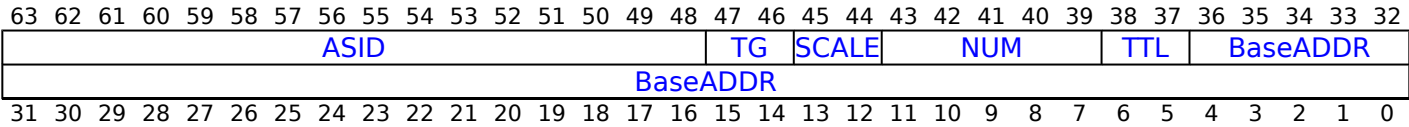
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVALE2OS, TLBI RVALE2OSNXS are UNDEFINED.

Attributes

TLBI RVALE2OS, TLBI RVALE2OSNXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]
When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	TheAll TTLentries hintto indicatesinvalidate levelare 1.Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	TheAll TTLentries hintto indicatesinvalidate levelare 2.Level 2 translation table entries.
0b11	TheAll TTLentries hintto indicatesinvalidate levelare 3.Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL2.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE2OS, TLBI RVALE2OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE2OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI RVALE3, TLBI RVALE3NXS, TLB Range Invalidate by VA, Last level, EL3

The TLBI RVALE3, TLBI RVALE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.~~walk.~~
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseADDR[29:12]$ is not equal to 00000000000000000000.
 - If $TTL == 10$ and $BaseADDR[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseADDR[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseADDR[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseADDR[28:16]$ is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

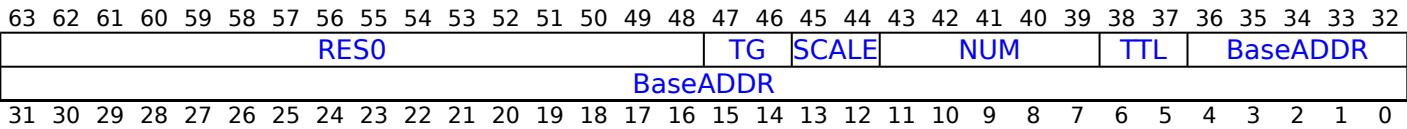
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE3, TLBI RVALE3NXS are UNDEFINED.

Attributes

TLBI RVALE3, TLBI RVALE3NXS is a 64-bit System instruction.

Field descriptions



Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]
When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE3, TLBI RVALE3NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0110	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBILevel_Last,
    TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE3NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBILevel_Last,
    TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RVALE3IS, TLBI RVALE3ISNXS, TLB Range Invalidate by VA, Last level, EL3, Inner Shareable

The TLBI RVALE3IS, TLBI RVALE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.~~walk.~~
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseAddr[29:12]$ is not equal to 00000000000000000000.
 - If $TTL == 10$ and $BaseAddr[20:12]$ is not equal to 000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseAddr[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseAddr[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseAddr[28:16]$ is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE3IS, TLBI RVALE3ISNXS are UNDEFINED.

Attributes

TLBI RVALE3IS, TLBI RVALE3ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG	SCALE	NUM					TTL			BaseADDR					
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing the TLBI RVALE3IS, TLBI RVALE3ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI RVALE3IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Last,
    TLBI_AllAttr, X[t, 64]);

```

TLBI RVALE3ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Last,
    TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI RVALE3OS, TLBI RVALE3OSNXS, TLB Range Invalidate by VA, Last level, EL3, Outer Shareable

The TLBI RVALE3OS, TLBI RVALE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 64-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseAddr[29:12]$ is not equal to 00000000000000000000.
 - If $TTL == 10$ and $BaseAddr[20:12]$ is not equal to 000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseAddr[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseAddr[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseAddr[28:16]$ is not equal to 0000000000000000.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented and FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI RVALE3OS, TLBI RVALE3OSNXS are UNDEFINED.

Attributes

TLBI RVALE3OS, TLBI RVALE3OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TG		SCALE		NUM				TTL		BaseADDR					
BaseADDR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate: **entries in the range that match the level described by the TTL hint.**

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.**
- Leaf-level entries in the range that match the level described by the TTL hint.**

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The All TTL entries hint to indicate invalidate level are 1. Level 1 translation table entries. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The All TTL entries hint to indicate invalidate level are 2. Level 2 translation table entries.
0b11	The All TTL entries hint to indicate invalidate level are 3. Level 3 translation table entries.

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL3.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Accesses to this instruction use the following encodings in the System instruction encoding space:

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH, TLBILevel_Last,
    TLBI AllAttr, X[t, 64]);

```

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH, TLBILevel_Last,
    TLBI_ExcludeXS, X[t, 64]);

```

(old)

htmldiff from-

(new)

TLBI VAAE1, TLBI VAAE1NXS, TLB Invalidate by VA, All ASID, EL1

The TLBI VAAE1, TLBI VAAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from any level of the translation table walk.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.

- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAAE1, TLBI VAAE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAAE1, TLBI VAAE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAAE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAAE1IS, TLBI VAAE1ISNXS, TLB Invalidate by VA, All ASID, EL1, Inner Shareable

The TLBI VAAE1IS, TLBI VAAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from any level of the translation table walk.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.

- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAAE1IS, TLBI VAAE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAAE1IS, TLBI VAAE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        endif
    endif
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    endif
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    endif
endif

```

TLBI VAAE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0011	0b011
------	-------	--------	--------	-------

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAAE1OS, TLBI VAAE1OSNXS, TLB Invalidate by VA, All ASID, EL1, Outer Shareable

The TLBI VAAE1OS, TLBI VAAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from any level of the translation table walk.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.

- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VAAE1OS, TLBI VAAE1OSNXS are UNDEFINED.

Attributes

TLBI VAAE1OS, TLBI VAAE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAAE1OS, TLBI VAAE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elseif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elseif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAAE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0001	0b011
------	-------	--------	--------	-------

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAALE1, TLBI VAALE1NXS, TLB Invalidate by VA, All ASID, Last level, EL1

The TLBI VAALE1, TLBI VAALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAALE1, TLBI VAALE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAALE1, TLBI VAALE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VAALE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b111


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAALE1IS, TLBI VAALE1ISNXS, TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

The TLBI VAALE1IS, TLBI VAALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAALE1IS, TLBI VAALE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAALE1IS, TLBI VAALE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        endif
    endif
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    endif
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    endif
endif

```

TLBI VAALE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0011	0b111
------	-------	--------	--------	-------

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TLBI VAALE1OS, TLBI VAALE1OSNXS, TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

The TLBI VAALE1OS, TLBI VAALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from the final level of the translation table walk.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VAALE1OS, TLBI VAALE1OSNXS are UNDEFINED.

Attributes

TLBI VAALE1OS, TLBI VAALE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAALE1OS, TLBI VAALE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elseif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elseif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VAALE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0001	0b111
------	-------	--------	--------	-------

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAE1, TLBI VAE1NXS, TLB Invalidate by VA, EL1

The TLBI VAE1, TLBI VAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE1, TLBI VAE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.

- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE1, TLBI VAE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAE1IS, TLBI VAE1ISNXS, TLB Invalidate by VA, EL1, Inner Shareable

The TLBI VAE1IS, TLBI VAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
- Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE1IS, TLBI VAE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE1IS, TLBI VAE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```


TLBI VAE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

html diff from-

(new)

TLBI VAE1OS, TLBI VAE1OSNXS, TLB Invalidate by VA, EL1, Outer Shareable

The TLBI VAE1OS, TLBI VAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
Or, if FEAT_D128 is implemented, and the entry is 128-bit a stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VAE1OS, TLBI VAE1OSNXS are UNDEFINED.

Attributes

TLBI VAE1OS, TLBI VAE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE1OS, TLBI VAE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
        endif
    endif
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    endif
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    endif
endif

```

TLBI VAE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
            end
        end
    end
end

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	html diff from-	(new)
-------	-----------------	-------

(old)

htmldiff from-

(new)

TLBI VAE2, TLBI VAE2NXS, TLB Invalidate by VA, EL2

The TLBI VAE2, TLBI VAE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE2, TLBI VAE2NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.

- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE2, TLBI VAE2NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE2NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0111	0b001


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAE2IS, TLBI VAE2ISNXS, TLB Invalidate by VA, EL2, Inner Shareable

The TLBI VAE2IS, TLBI VAE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from any level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE2IS, TLBI VAE2ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.

- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE2IS, TLBI VAE2ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE2ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TLBI VAE2OS, TLBI VAE2OSNXS, TLB Invalidate by VA, EL2, Outer Shareable

The TLBI VAE2OS, TLBI VAE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from any level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VAE2OS, TLBI VAE2OSNXS are UNDEFINED.

Attributes

TLBI VAE2OS, TLBI VAE2OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE2OS, TLBI VAE2OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE2OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0001	0b001


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAE3, TLBI VAE3NXS, TLB Invalidate by VA, EL3

The TLBI VAE3, TLBI VAE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk.

Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE3, TLBI VAE3NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																TTL			VA[55:12]													
VA[55:12]																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE3, TLBI VAE3NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

```
TLBI VAE3NXS{, <Xt>}
```

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0111	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VAE3IS, TLBI VAE3ISNXS, TLB Invalidate by VA, EL3, Inner Shareable

The TLBI VAE3IS, TLBI VAE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk.

Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VAE3IS, TLBI VAE3ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE3IS, TLBI VAE3ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE3IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE3ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TLBI VAE3OS, TLBI VAE3OSNXS, TLB Invalidate by VA, EL3, Outer Shareable

The TLBI VAE3OS, TLBI VAE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk.

Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VAE3OS, TLBI VAE3OSNXS are UNDEFINED.

Attributes

TLBI VAE3OS, TLBI VAE3OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE3OS, TLBI VAE3OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VAE3OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI VAE3OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0001	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE1, TLBI VALE1NXS, TLB Invalidate by VA, Last level, EL1

The TLBI VALE1, TLBI VALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VALE1, TLBI VALE1NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.

- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE1, TLBI VALE1NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
                elsif PSTATE.EL == EL3 then
                    if HCR_EL2.<E2H,TGE> == '11' then
                        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
                    else
                        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE1IS, TLBI VALE1ISNXS, TLB Invalidate by VA, Last level, EL1, Inner Shareable

The TLBI VALE1IS, TLBI VALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VALE1IS, TLBI VALE1ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																TTL				VA[55:12]													
VA[55:12]																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44] When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE1IS, TLBI VALE1ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        endif
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        endif
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        endif
    endif
endif

```

TLBI VALE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            end
        end
    end
end

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	html diff from-	(new)
-------	-----------------	-------

(old)

htmldiff from-

(new)

TLBI VALE1OS, TLBI VALE1OSNXS, TLB Invalidate by VA, Last level, EL1, Outer Shareable

The TLBI VALE1OS, TLBI VALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if SCR_EL3.EEL2==1, then:

- A PE with SCR_EL3.EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==0.
 - A PE with SCR_EL3.EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with SCR_EL3.EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VALE1OS, TLBI VALE1OSNXS are UNDEFINED.

Attributes

TLBI VALE1OS, TLBI VALE1OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
																VA[55:12]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE1OS, TLBI VALE1OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
                else
                    AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE1OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
            else
                AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

html diff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE2, TLBI VALE2NXS, TLB Invalidate by VA, Last level, EL2

The TLBI VALE2, TLBI VALE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VALE2, TLBI VALE2NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
																VA[55:12]															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]**When HCR_EL2.E2H == 1:**

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE2, TLBI VALE2NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE2NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE2IS, TLBI VALE2ISNXS, TLB Invalidate by VA, Last level, EL2, Inner Shareable

The TLBI VALE2IS, TLBI VALE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VALE2IS, TLBI VALE2ISNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.

- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE2IS, TLBI VALE2ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE2ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0011	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE2OS, TLBI VALE2OSNXS, TLB Invalidate by VA, Last level, EL2, Outer Shareable

The TLBI VALE2OS, TLBI VALE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.
- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.
- If HCR_EL2.E2H == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VALE2OS, TLBI VALE2OSNXS are UNDEFINED.

Attributes

TLBI VALE2OS, TLBI VALE2OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ASID																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ASID, bits [63:48]**When HCR_EL2.E2H == 1:**

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE2OS, TLBI VALE2OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE2OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE3, TLBI VALE3NXS, TLB Invalidate by VA, Last level, EL3

The TLBI VALE3, TLBI VALE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk.

Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VALE3, TLBI VALE3NXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE3, TLBI VALE3NXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE3NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TLBI VALE3IS, TLBI VALE3ISNXS characteristics are:

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a **64-bit** stage 1 translation table entry, from the final level of the translation table walk.
Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is **0b00**.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

There are no configuration notes.

TLBI VALE3IS, TLBI VALE3ISNXS is a 64-bit System instruction.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE3IS, TLBI VALE3ISNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE3IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0011	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE3ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0011	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VALE3OS, TLBI VALE3OSNXS, TLB Invalidate by VA, Last level, EL3, Outer Shareable

The TLBI VALE3OS, TLBI VALE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk.

Or, if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.

- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VALE3OS, TLBI VALE3OSNXS are UNDEFINED.

Attributes

TLBI VALE3OS, TLBI VALE3OSNXS is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																TTL				VA[55:12]											
VA[55:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE3OS, TLBI VALE3OSNXS instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VALE3OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_OSH,
    TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

TLBI VALE3OSNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, VMID[], Shareability_OSH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TLBI VMALLE1, TLBI VMALLE1NXS, TLB Invalidate by VMID, All at stage 1, EL1

The TLBI VMALLE1, TLBI VMALLE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VMALLE1, TLBI VMALLE1NXS is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the TLBI VMALLE1, TLBI VMALLE1NXS instruction

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VMALLE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIVMALLE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBI_AllAttr);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBI_ExcludeXS);
            else
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBI_AllAttr);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBI_AllAttr);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBI_AllAttr);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBI_AllAttr);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBI_AllAttr);

```

TLBI VMALLE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b000

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVMALLE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_ExcludeXS);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBI_ExcludeXS);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBI_ExcludeXS);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBI_ExcludeXS);
            else
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_ExcludeXS);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBI VMALLE1IS, TLBI VMALLE1ISNXS, TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

The TLBI VMALLE1IS, TLBI VMALLE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

There are no configuration notes.

Attributes

TLBI VMALLE1IS, TLBI VMALLE1ISNXS is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the TLBI VMALLE1IS, TLBI VMALLE1ISNXS instruction

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VMALLE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVMALLE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr);
        endif
    endif
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBI_AllAttr);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_AllAttr);
    endif
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBI_AllAttr);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_AllAttr);
    endif
endif

```

TLBI VMALLE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b000	0b1001	0b0011	0b000
------	-------	--------	--------	-------

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVMALLE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBI_ExcludeXS);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_ExcludeXS);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBI_ExcludeXS);
            else
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBI_ExcludeXS);
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TLBI VMALLE1OS, TLBI VMALLE1OSNXS, TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable

The TLBI VMALLE1OS, TLBI VMALLE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_TLBIOS is implemented. Otherwise, direct accesses to TLBI VMALLE1OS, TLBI VMALLE1OSNXS are UNDEFINED.

Attributes

TLBI VMALLE1OS, TLBI VMALLE1OSNXS is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing the TLBI VMALLE1OS, TLBI VMALLE1OSNXS instruction

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE whether:

- The instruction is UNDEFINED.
- The instruction behaves as if the Rt field is set to 0b11111.

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBI VMALLE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIVMALLE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_AllAttr);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBI_AllAttr);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_AllAttr);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBI_AllAttr);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBI_AllAttr);

```

TLBI VMALLE10SNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b000

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVMALLE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBI_ExcludeXS);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBI_ExcludeXS);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBI_ExcludeXS);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBI_ExcludeXS);
            else
                AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBI_ExcludeXS);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP IPAS2E1, TLBIP IPAS2E1NXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1

The TLBIP IPAS2E1, TLBIP IPAS2E1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk.
- Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2E1, TLBIP IPAS2E1NXS are UNDEFINED.

Attributes

TLBIP IPAS2E1, TLBIP IPAS2E1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	
RES0																IPA[55:12]																
IPA[55:12]																																
NS	RES0															TTL					RES0											
RES0																																

Bits [127:108]

Reserved, RES0.

IPA[55:12], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]

When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2E1, TLBIP IPAS2E1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2E1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP IPAS2E1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

The TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk.
Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS are UNDEFINED.

Attributes

TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		IPA[55:12]																	
IPA[55:12]																																			
NS	RES0																	TTL	RES0																
RES0																																			

Bits [127:108]

Reserved, RES0.

IPA[55:12], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44] When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2E1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
```

TLBIP IPAS2E1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0000	0b001

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

The TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk.
Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS are UNDEFINED.

Attributes

TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS is a 128-bit System instruction.

Field descriptions

12712612512412312212112011911811711611511411311211111010910810710610510410310210110099989796	
RES0	IPA[51:48]
IPA[51:48]	
NS	RES0TTLRES0
RES0	

Bits [127:108]

Reserved, RES0.

IPA[51:48], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2E1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP IPAS2E10SNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b000

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file htmldiff from- (new)

TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1

The TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS are UNDEFINED.

Attributes

TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS is a 128-bit System instruction.

Field descriptions

12712612512412312212112011911811711611511411311211111010910810710610510410310210110099989796	
RES0	IPA[55:12]
IPA[55:12]	
NS	RES0TTLRES0
RES0	

Bits [127:108]

Reserved, RES0.

IPA[55:12], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2LE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP IPAS2LE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b101

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file htmldiff from- (new)

TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

The TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS are UNDEFINED.

Attributes

TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS is a 128-bit System instruction.

Field descriptions

12712612512412312212112011911811711611511411311211111010910810710610510410310210110099989796	
RES0	IPA[55:12]
IPA[55:12]	
NSRES0	TTLRES0
RES0	

Bits [127:108]

Reserved, RES0.

IPA[55:12], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2LE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
```

TLBIP IPAS2LE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0000	0b101

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
    endif
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    endif
endif
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file	htmldiff from-	(new)
-------------	----------------	-------

TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

The TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 2 only translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS are UNDEFINED.

Attributes

TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS is a 128-bit System instruction.

Field descriptions

12712612512412312212112011911811711611511411311211111010910810710610510410310210110099989796	
RES0	IPA[51:48]
IPA[51:48]	
NS	RES0TTLRES0
RES0	

Bits [127:108]

Reserved, RES0.

IPA[51:48], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP IPAS2LE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP IPAS2LE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b100

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1

The TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS are UNDEFINED.

Attributes

TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
NS	RES0														TG	SCALE	NUM				TTL	RES0													
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63]

When FEAT_RME is implemented:

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and [SCR_EL3](#).{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2E1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2E1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b010

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

The TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS are UNDEFINED.

Attributes

TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
NS	RES0															TG	SCALE	NUM				TTL	RES0												
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2E1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2E1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0000	0b010

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

The TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 2 only translation table entry, from any level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS are UNDEFINED.

Attributes

TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
NS	RES0														TG	SCALE	NUM				TTL		RES0												
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2E1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2E1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

The TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- Or if FEAT_D128 is implemented, and the entry is a 64-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS are UNDEFINED.

Attributes

TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
NS	RES0														TG	SCALE	NUM				TTL		RES0								
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2LE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2LE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b110

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

The TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.

Or if FEAT_D128 is implemented, and the entry is a 64-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.

- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS are UNDEFINED.

Attributes

TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
NS	RES0														TG	SCALE	NUM				TTL		RES0												
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2LE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RIPAS2LE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0000	0b110

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

The TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS characteristics are:

Purpose

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- Or if FEAT_D128 is implemented, and the entry is a 64-bit stage 2 only translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- If FEAT_RME is implemented, one of the following applies:
 - [SCR_EL3](#).{NSE, NS} is {0, 0} and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {0, 1} and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
 - [SCR_EL3](#).{NSE, NS} is {1, 1} and the entry would be required to translate any IPA in the specified address range using the Realm EL1&0 translation regime.
- If FEAT_RME is not implemented, one of the following applies:
 - [SCR_EL3](#).NS is 0 and the entry would be required to translate any IPA in the specified address range using the Secure EL1&0 translation regime.
 - [SCR_EL3](#).NS is 1 and the entry would be required to translate any IPA in the specified address range using the Non-secure EL1&0 translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
- A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS are UNDEFINED.

Attributes

TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
NS	RES0													TG	SCALE	NUM					TTL			RES0											
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

NS, bit [63] When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is RES0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is RES0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is RES0.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RIPAS2LE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
```

TLBIP RIPAS2LE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b111

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

TLBIP RVAAE1, TLBIP RVAAE1NXS, TLB Range Invalidate by VA, All ASID, EL1

The TLBIP RVAAE1, TLBIP RVAAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAAE1, TLBIP RVAAE1NXS are UNDEFINED.

Attributes

TLBIP RVAAE1, TLBIP RVAAE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	
RES0																BaseADDR[55:12]																
BaseADDR[55:12]																																
RES0																TG	SCALE	NUM				TTL	RES0									
RES0																																

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAAE1, TLBIP RVAAE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAAE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAAE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end
end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS, TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

The TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS are UNDEFINED.

Attributes

TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAAE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAAE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS, TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

The TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS are UNDEFINED.

Attributes

TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL	RES0								
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAAE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAAE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAALE1, TLBIP RVAALE1NXS, TLB Range Invalidate by VA, All ASID, Last level, EL1

The TLBIP RVAALE1, TLBIP RVAALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
Or the entry is a 64-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAALE1, TLBIP RVAALE1NXS are UNDEFINED.

Attributes

TLBIP RVAALE1, TLBIP RVAALE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
RES0														TG	SCALE	NUM				TTL	RES0														
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAALE1, TLBIP RVAALE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAALE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAALE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end
end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

The TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
Or the entry is a 64-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS are UNDEFINED.

Attributes

TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
RES0																TG	SCALE	NUM				TTL		RES0											
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAALE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAALE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

The TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
Or the entry is a 64-bit stage 1 translation table entry, from the leaf level of the translation table walk, if TTL is 0b00.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS are UNDEFINED.

Attributes

TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAALE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAALE10SNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE1, TLBIP RVAE1NXS, TLB Range Invalidate by VA, EL1

The TLBIP RVAE1, TLBIP RVAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE1, TLBIP RVAE1NXS are UNDEFINED.

Attributes

TLBIP RVAE1, TLBIP RVAE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE1, TLBIP RVAE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE1IS, TLBIP RVAE1ISNXS, TLB Range Invalidate by VA, EL1, Inner Shareable

The TLBIP RVAE1IS, TLBIP RVAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE1IS, TLBIP RVAE1ISNXS are UNDEFINED.

Attributes

TLBIP RVAE1IS, TLBIP RVAE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
ASID														TG	SCALE	NUM						TTL	RES0												
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE1IS, TLBIP RVAE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE1OS, TLBIP RVAE1OSNXS, TLB Range Invalidate by VA, EL1, Outer Shareable

The TLBIP RVAE1OS, TLBIP RVAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE1OS, TLBIP RVAE1OSNXS are UNDEFINED.

Attributes

TLBIP RVAE1OS, TLBIP RVAE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
ASID														TG	SCALE	NUM						TTL	RES0												
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE1OS, TLBIP RVAE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE10SNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVAE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE2, TLBIP RVAE2NXS, TLB Range Invalidate by VA, EL2

The TLBIP RVAE2, TLBIP RVAE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE2, TLBIP RVAE2NXS are UNDEFINED.

Attributes

TLBIP RVAE2, TLBIP RVAE2NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
ASID														TG	SCALE	NUM				TTL	RES0														
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE2, TLBIP RVAE2NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE2{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    
```

TLBIP RVAE2NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE2IS, TLBIP RVAE2ISNXS, TLB Range Invalidate by VA, EL2, Inner Shareable

The TLBIP RVAE2IS, TLBIP RVAE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[\text{BaseADDR} \leq \text{VA} < \text{BaseADDR} + ((\text{NUM} + 1) * 2^{(5 * \text{SCALE} + 1)} * \text{Translation_Granule_Size})]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE2IS, TLBIP RVAE2ISNXS are UNDEFINED.

Attributes

TLBIP RVAE2IS, TLBIP RVAE2ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE2IS, TLBIP RVAE2ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE2IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE2ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0010	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE2OS, TLBIP RVAE2OSNXS, TLB Range Invalidate by VA, EL2, Outer Shareable

The TLBIP RVAE2OS, TLBIP RVAE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE2OS, TLBIP RVAE2OSNXS are UNDEFINED.

Attributes

TLBIP RVAE2OS, TLBIP RVAE2OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID												TG	SCALE	NUM				TTL		RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE2OS, TLBIP RVAE2OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE2OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    
```

TLBIP RVAE2OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0101	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVAE3, TLBIP RVAE3NXS, TLB Range Invalidate by VA, EL3

The TLBIP RVAE3, TLBIP RVAE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE3, TLBIP RVAE3NXS are UNDEFINED.

Attributes

TLBIP RVAE3, TLBIP RVAE3NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE3, TLBIP RVAE3NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE3{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0110	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBILevel_Any,
    TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE3NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0110	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH, TLBILevel_Any,
    TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP RVAE3IS, TLBIP RVAE3ISNXS, TLB Range Invalidate by VA, EL3, Inner Shareable

The TLBIP RVAE3IS, TLBIP RVAE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE3IS, TLBIP RVAE3ISNXS are UNDEFINED.

Attributes

TLBIP RVAE3IS, TLBIP RVAE3ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE3IS, TLBIP RVAE3ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE3IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0010	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Any,
    TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE3ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0010	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH, TLBIlevel_Any,
    TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP RVAE3OS, TLBIP RVAE3OSNXS, TLB Range Invalidate by VA, EL3, Outer Shareable

The TLBIP RVAE3OS, TLBIP RVAE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVAE3OS, TLBIP RVAE3OSNXS are UNDEFINED.

Attributes

TLBIP RVAE3OS, TLBIP RVAE3OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96		
RES0																BaseADDR[55:12]																	
BaseADDR[55:12]																																	
RES0																TG	SCALE	NUM				TTL				RES0							
RES0																																	

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVAE3OS, TLBIP RVAE3OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVAE3OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0101	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH, TLBILevel_Any,
    TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVAE3OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0101	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH, TLBILevel_Any,
    TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE1, TLBIP RVALE1NXS, TLB Range Invalidate by VA, Last level, EL1

The TLBIP RVALE1, TLBIP RVALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE1, TLBIP RVALE1NXS are UNDEFINED.

Attributes

TLBIP RVALE1, TLBIP RVALE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID												TG	SCALE	NUM				TTL	RES0												
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.

- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE1, TLBIP RVALE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```


TLBIP RVALE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE1IS, TLBIP RVALE1ISNXS, TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

The TLBIP RVALE1IS, TLBIP RVALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE1IS, TLBIP RVALE1ISNXS are UNDEFINED.

Attributes

TLBIP RVALE1IS, TLBIP RVALE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE1IS, TLBIP RVALE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE1OS, TLBIP RVALE1OSNXS, TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

The TLBIP RVALE1OS, TLBIP RVALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseAddr \leq VA < BaseAddr + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate any of the VAs in the specified address range using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE1OS, TLBIP RVALE1OSNXS are UNDEFINED.

Attributes

TLBIP RVALE1OS, TLBIP RVALE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE1OS, TLBIP RVALE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIRVALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE10SNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE2, TLBIP RVALE2NXS, TLB Range Invalidate by VA, Last level, EL2

The TLBIP RVALE2, TLBIP RVALE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[\text{BaseADDR} \leq \text{VA} < \text{BaseADDR} + ((\text{NUM} + 1) * 2^{(5 * \text{SCALE} + 1)} * \text{Translation_Granule_Size})]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE2, TLBIP RVALE2NXS are UNDEFINED.

Attributes

TLBIP RVALE2, TLBIP RVALE2NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		BaseADDR[55:12]																	
BaseADDR[55:12]																																			
ASID														TG	SCALE	NUM				TTL	RES0														
RES0																																			

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]
When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE2, TLBIP RVALE2NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE2{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE2NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE2IS, TLBIP RVALE2ISNXS, TLB Range Invalidate by VA, Last level, EL2, Inner Shareable

The TLBIP RVALE2IS, TLBIP RVALE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE2IS, TLBIP RVALE2ISNXS are UNDEFINED.

Attributes

TLBIP RVALE2IS, TLBIP RVALE2ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	
RES0																BaseADDR[55:12]																
BaseADDR[55:12]																																
ASID																TG	SCALE	NUM				TTL	RES0									
RES0																																

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE2IS, TLBIP RVALE2ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE2IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE2ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE2OS, TLBIP RVALE2OSNXS, TLB Range Invalidate by VA, Last level, EL2, Outer Shareable

The TLBIP RVALE2OS, TLBIP RVALE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.

Or the entry is a 64-bit stage 1 translation table entry, if TTL is 0b00.

- The entry would be used to translate any VA in the range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$ using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE2OS, TLBIP RVALE2OSNXS are UNDEFINED.

Attributes

TLBIP RVALE2OS, TLBIP RVALE2OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
ASID																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE2OS, TLBIP RVALE2OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE2OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE2OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP RVALE3, TLBIP RVALE3NXS, TLB Range Invalidate by VA, Last level, EL3

The TLBIP RVALE3, TLBIP RVALE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE3, TLBIP RVALE3NXS are UNDEFINED.

Attributes

TLBIP RVALE3, TLBIP RVALE3NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	
RES0																BaseADDR[55:12]																
BaseADDR[55:12]																																
RES0																TG	SCALE	NUM					TTL			RES0						
RES0																																

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE3, TLBIP RVALE3NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE3{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0110	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE3NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0110	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP RVALE3IS, TLBIP RVALE3ISNXS, TLB Range Invalidate by VA, Last level, EL3, Inner Shareable

The TLBIP RVALE3IS, TLBIP RVALE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE3IS, TLBIP RVALE3ISNXS are UNDEFINED.

Attributes

TLBIP RVALE3IS, TLBIP RVALE3ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE3IS, TLBIP RVALE3ISNXXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE3IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0010	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE3ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0010	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_ISH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP RVALE3OS, TLBIP RVALE3OSNXS, TLB Range Invalidate by VA, Last level, EL3, Outer Shareable

The TLBIP RVALE3OS, TLBIP RVALE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk up to the level indicated in the TTL hint.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL is 0b00.
- The entry would be used to translate any of the VAs in the specified address range using the EL3 translation regime.
- The entry is within the address range determined by the formula $[BaseADDR \leq VA < BaseADDR + ((NUM + 1) * 2^{(5 * SCALE + 1)} * Translation_Granule_Size)]$.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For 128-bit translation table entry, the range of addresses invalidated is UNPREDICTABLE when Block or Page size corresponding to TTL and TG, for the translation system is not aligned.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP RVALE3OS, TLBIP RVALE3OSNXS are UNDEFINED.

Attributes

TLBIP RVALE3OS, TLBIP RVALE3OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																BaseADDR[55:12]															
BaseADDR[55:12]																															
RES0																TG	SCALE	NUM				TTL		RES0							
RES0																															

Bits [127:108]

Reserved, RES0.

BaseADDR[55:12], bits [107:64]

The starting address for the range of the maintenance instructions. This field is BaseADDR[55:12] for all translation granules.

Bits [63:48]

Reserved, RES0.

TG, bits [47:46]

Translation granule size.

TG	Meaning
0b00	Reserved.
0b01	4K translation granule.
0b10	16K translation granule.
0b11	64K translation granule.

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

TTL	Meaning
0b00	The entries in the range can be using any level for the translation table entries.
0b01	The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.
0b10	The TTL hint indicates level 2.
0b11	The TTL hint indicates level 3.

Bits [36:0]

Reserved, RES0.

Executing TLBIP RVALE3OS, TLBIP RVALE3OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP RVALE3OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0101	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH,
    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP RVALE3OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0101	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_RVA(SecurityStateAtEL(EL3), Regime_EL3, VMID[], Shareability_0SH,
    TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAAE1, TLBIP VAAE1NXS, TLB Invalidate Pair by VA, All ASID, EL1

The TLBIP VAAE1, TLBIP VAAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAAE1, TLBIP VAAE1NXS are UNDEFINED.

Attributes

TLBIP VAAE1, TLBIP VAAE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level.
0b01xx	In this case, TTL<1:0> is RES0. The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAAE1, TLBIP VAAE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAAE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGITR_EL2.TLBIVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                elsif PSTATE.EL == EL3 then
                    if HCR_EL2.<E2H,TGE> == '11' then
                        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                    else
                        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAAE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAAE1IS, TLBIP VAAE1ISNXS, TLB Invalidate Pair by VA, All ASID, EL1, Inner Shareable

The TLBIP VAAE1IS, TLBIP VAAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAAE1IS, TLBIP VAAE1ISNXS are UNDEFINED.

Attributes

TLBIP VAAE1IS, TLBIP VAAE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
RES0																		TTL				RES0													
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAAE1IS, TLBIP VAAE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAAE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAAE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b011


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAAE1OS, TLBIP VAAE1OSNXS, TLB Invalidate Pair by VA, All ASID, EL1, Outer Shareable

The TLBIP VAAE1OS, TLBIP VAAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
- Or the entry is a 64-bit stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAAE1OS, TLBIP VAAE1OSNXS are UNDEFINED.

Attributes

TLBIP VAAE1OS, TLBIP VAAE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
RES0																TTL				RES0															
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAAE1OS, TLBIP VAAE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAAE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAAE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b011

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAALE1, TLBIP VAALE1NXS, TLB Invalidate Pair by VA, All ASID, Last level, EL1

The TLBIP VAALE1, TLBIP VAALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAALE1, TLBIP VAALE1NXS are UNDEFINED.

Attributes

TLBIP VAALE1, TLBIP VAALE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level.
0b01xx	In this case, TTL<1:0> is RES0. The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAALE1, TLBIP VAALE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAALE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                elsif PSTATE.EL == EL3 then
                    if HCR_EL2.<E2H,TGE> == '11' then
                        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                    else
                        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAALE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            end
        end
    end
end

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAALE1IS, TLBIP VAALE1ISNXS, TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Inner Shareable

The TLBIP VAALE1IS, TLBIP VAALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAALE1IS, TLBIP VAALE1ISNXS are UNDEFINED.

Attributes

TLBIP VAALE1IS, TLBIP VAALE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAALE1IS, TLBIP VAALE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAALE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAALE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAALE1OS, TLBIP VAALE1OSNXS, TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Outer Shareable

The TLBIP VAALE1OS, TLBIP VAALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAALE1OS, TLBIP VAALE1OSNXS are UNDEFINED.

Attributes

TLBIP VAALE1OS, TLBIP VAALE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
RES0																		TTL				RES0													
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAALE1OS, TLBIP VAALE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAALE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAALE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b111

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE1, TLBIP VAE1NXS, TLB Invalidate Pair by VA, EL1

The TLBIP VAE1, TLBIP VAE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE1, TLBIP VAE1NXS are UNDEFINED.

Attributes

TLBIP VAE1, TLBIP VAE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE1, TLBIP VAE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
    && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b001


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVAE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE1IS, TLBIP VAE1ISNXS, TLB Invalidate Pair by VA, EL1, Inner Shareable

The TLBIP VAE1IS, TLBIP VAE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE1IS, TLBIP VAE1ISNXS are UNDEFINED.

Attributes

TLBIP VAE1IS, TLBIP VAE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
ASID																TTL				RES0															
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE1IS, TLBIP VAE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE1OS, TLBIP VAE1OSNXS, TLB Invalidate Pair by VA, EL1, Outer Shareable

The TLBIP VAE1OS, TLBIP VAE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE1OS, TLBIP VAE1OSNXS are UNDEFINED.

Attributes

TLBIP VAE1OS, TLBIP VAE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
ASID																TTL				RES0															
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE1OS, TLBIP VAE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE2, TLBIP VAE2NXS, TLB Invalidate Pair by VA, EL2

The TLBIP VAE2, TLBIP VAE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE2, TLBIP VAE2NXS are UNDEFINED.

Attributes

TLBIP VAE2, TLBIP VAE2NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE2, TLBIP VAE2NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE2{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE2NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0111	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE2IS, TLBIP VAE2ISNXS, TLB Invalidate Pair by VA, EL2, Inner Shareable

The TLBIP VAE2IS, TLBIP VAE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE2IS, TLBIP VAE2ISNXS are UNDEFINED.

Attributes

TLBIP VAE2IS, TLBIP VAE2ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE2IS, TLBIP VAE2ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE2IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE2ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VAE2OS, TLBIP VAE2OSNXS, TLB Invalidate Pair by VA, EL2, Outer Shareable

The TLBIP VAE2OS, TLBIP VAE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be required to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from any level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is from a level of the translation table walk above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE2OS, TLBIP VAE2OSNXS are UNDEFINED.

Attributes

TLBIP VAE2OS, TLBIP VAE2OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

When HCR_EL2.E2H == 1:

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE2OS, TLBIP VAE2OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE2OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE2OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0001	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VAE3, TLBIP VAE3NXS, TLB Invalidate Pair by VA, EL3

The TLBIP VAE3, TLBIP VAE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE3, TLBIP VAE3NXS are UNDEFINED.

Attributes

TLBIP VAE3, TLBIP VAE3NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE3, TLBIP VAE3NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE3{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0111	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE3NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0111	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_NSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VAE3IS, TLBIP VAE3ISNXS, TLB Invalidate Pair by VA, EL3, Inner Shareable

The TLBIP VAE3IS, TLBIP VAE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE3IS, TLBIP VAE3ISNXS are UNDEFINED.

Attributes

TLBIP VAE3IS, TLBIP VAE3ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE3IS, TLBIP VAE3ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE3IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0011	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_ISH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE3ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0011	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_ISH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VAE3OS, TLBIP VAE3OSNXS, TLB Invalidate Pair by VA, EL3, Outer Shareable

The TLBIP VAE3OS, TLBIP VAE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from any level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAE3OS, TLBIP VAE3OSNXS are UNDEFINED.

Attributes

TLBIP VAE3OS, TLBIP VAE3OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VAE3OS, TLBIP VAE3OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VAE3OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0001	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_OSH,
    TLBILevel_Any, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VAE3OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0001	0b001

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_OSH,
    TLBILevel_Any, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VALE1, TLBIP VALE1NXS, TLB Invalidate Pair by VA, Last level, EL1

The TLBIP VALE1, TLBIP VALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE1, TLBIP VALE1NXS are UNDEFINED.

Attributes

TLBIP VALE1, TLBIP VALE1NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE1, TLBIP VALE1NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        else
            if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
            && HCRX_EL2.FnXS == '1' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL2 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                elsif PSTATE.EL == EL3 then
                    if HCR_EL2.<E2H,TGE> == '11' then
                        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                        TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                    else
                        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                        TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
    HFGITR_EL2.TLBIVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VALE1IS, TLBIP VALE1ISNXS, TLB Invalidate Pair by VA, Last level, EL1, Inner Shareable

The TLBIP VALE1IS, TLBIP VALE1ISNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE1IS, TLBIP VALE1ISNXS are UNDEFINED.

Attributes

TLBIP VALE1IS, TLBIP VALE1ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE1IS, TLBIP VALE1ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE1IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE1ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b101


```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVALE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VALE1OS, TLBIP VALE1OSNXS, TLB Invalidate Pair by VA, Last level, EL1, Outer Shareable

The TLBIP VALE1OS, TLBIP VALE1OSNXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- When EL2 is implemented and enabled in the current Security state:
 - If [HCR_EL2](#).{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
 - If [HCR_EL2](#).{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.
- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The Security state is indicated by the value of [SCR_EL3](#).NS if FEAT_RME is not implemented, or [SCR_EL3](#).{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Note

When a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass a VMID argument if [SCR_EL3](#).EEL2==1, then:

- A PE with [SCR_EL3](#).EEL2==1 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==0.
 - A PE with [SCR_EL3](#).EEL2==0 is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3](#).EEL2==1.
 - A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.
-

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE1OS, TLBIP VALE1OSNXS are UNDEFINED.

Attributes

TLBIP VALE1OS, TLBIP VALE1OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96				
RES0																		VA[55:12]																	
VA[55:12]																																			
ASID																TTL				RES0															
RES0																																			

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE1OS, TLBIP VALE1OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGITR_EL2.TLBIVALE1OS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL2 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
            elsif PSTATE.EL == EL3 then
                if HCR_EL2.<E2H,TGE> == '11' then
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
                else
                    AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
                    TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.TTLB0S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& IsFeatureImplemented(FEAT_HCX) && (!IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVALE10S == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else
            AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        elsif PSTATE.EL == EL3 then
            if HCR_EL2.<E2H,TGE> == '11' then
                AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
            else
                AArch64.TLBIP_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TLBIP VALE2, TLBIP VALE2NXS, TLB Invalidate Pair by VA, Last level, EL2

The TLBIP VALE2, TLBIP VALE2NXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE2, TLBIP VALE2NXS are UNDEFINED.

Attributes

TLBIP VALE2, TLBIP VALE2NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																RES0															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]**When HCR_EL2.E2H == 1:**

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE2, TLBIP VALE2NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE2{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE2NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VALE2IS, TLBIP VALE2ISNXS, TLB Invalidate Pair by VA, Last level, EL2, Inner Shareable

The TLBIP VALE2IS, TLBIP VALE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.
- Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE2IS, TLBIP VALE2ISNXS are UNDEFINED.

Attributes

TLBIP VALE2IS, TLBIP VALE2ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE2IS, TLBIP VALE2ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE2IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE2ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0011	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VALE2OS, TLBIP VALE2OSNXS, TLB Invalidate Pair by VA, Last level, EL2, Outer Shareable

The TLBIP VALE2OS, TLBIP VALE2OSNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry.
- Or the entry is a 64-bit stage 1 translation table entry, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the [HCR_EL2.E2H](#) bit, for the Security state.
- If [HCR_EL2.E2H](#) == 0, the entry is from the final level of the translation table walk.
- If [HCR_EL2.E2H](#) == 1, one of the following applies:
 - The entry is a global entry from the final level of the translation table walk.
 - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of [SCR_EL3.NS](#) if FEAT_RME is not implemented, or [SCR_EL3.{NSE, NS}](#) if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE2OS, TLBIP VALE2OSNXS are UNDEFINED.

Attributes

TLBIP VALE2OS, TLBIP VALE2OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
ASID								TTL				RES0																			
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

ASID, bits [63:48]**When HCR_EL2.E2H == 1:**

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

Otherwise:

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE2OS, TLBIP VALE2OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE2OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE2OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL20, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VALE3, TLBIP VALE3NXS, TLB Invalidate Pair by VA, Last level, EL3

The TLBIP VALE3, TLBIP VALE3NXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE3, TLBIP VALE3NXS are UNDEFINED.

Attributes

TLBIP VALE3, TLBIP VALE3NXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0																TTL				RES0											
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE3, TLBIP VALE3NXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE3{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0111	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_NSH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE3NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0111	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_NSH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VALE3IS, TLBIP VALE3ISNXS, TLB Invalidate Pair by VA, Last level, EL3, Inner Shareable

The TLBIP VALE3IS, TLBIP VALE3ISNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE3IS, TLBIP VALE3ISNXS are UNDEFINED.

Attributes

TLBIP VALE3IS, TLBIP VALE3ISNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
RES0																VA[55:12]															
VA[55:12]																															
RES0								TTL				RES0																			
RES0																															

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]**When FEAT_TTL is implemented:**

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE3IS, TLBIP VALE3ISNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE3IS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
------------	------------	------------	------------	------------

0b01	0b110	0b1000	0b0011	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_ISH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE3ISNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0011	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_ISH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TLBIP VALE3OS, TLBIP VALE3OSNXS, TLB Invalidate Pair by VA, Last level, EL3, Outer Shareable

The TLBIP VALE3OS, TLBIP VALE3OSNXS characteristics are:

Purpose

If EL3 is implemented, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.
Or the entry is 64-bit a stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is 0b00.
- The entry would be used to translate the specified VA using the EL3 translation regime.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VALE3OS, TLBIP VALE3OSNXS are UNDEFINED.

Attributes

TLBIP VALE3OS, TLBIP VALE3OSNXS is a 128-bit System instruction.

Field descriptions

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96						
RES0																		VA[55:12]																			
VA[55:12]																																					
RES0																		TTL		RES0																	
RES0																																					

Bits [127:108]

Reserved, RES0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Bits [63:48]

Reserved, RES0.

TTL, bits [47:44]

When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.
0b01xx	The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.
0b10xx	The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3.
0b11xx	The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, RES0.

Bits [43:0]

Reserved, RES0.

Executing TLBIP VALE3OS, TLBIP VALE3OSNXS

Accesses to this instruction use the following encodings in the System instruction encoding space:

TLBIP VALE3OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b01	0b110	0b1000	0b0001	0b101
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_OSH,
    TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);

```

TLBIP VALE3OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1001	0b0001	0b101

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.TLBIP_VA(SecurityStateAtEL(EL3), Regime_EL3, VMID_NONE, Shareability_OSH,
    TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);

```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TPIDR2_EL0, EL0 Read/Write Software Thread ID Register 2

The TPIDR2_EL0 characteristics are:

Purpose

Provides a location where SME-aware software executing at EL0 can store thread identifying information, for context management purposes.

The PE makes no use of this register.

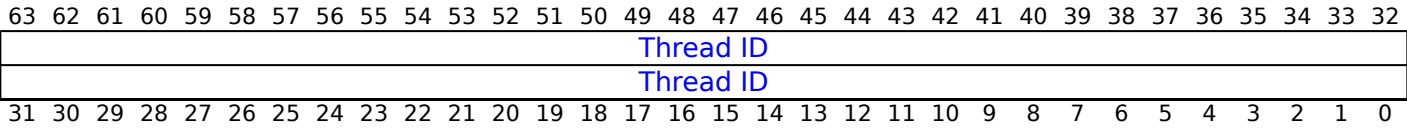
Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to TPIDR2_EL0 are UNDEFINED.

Attributes

TPIDR2_EL0 is a 64-bit register.

Field descriptions



Bits [63:0]

Thread identifying information stored by software running at this Exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TPIDR2_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TPIDR2_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
        UNDEFINED;
    elseif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnTP2 == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnTP2 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.nTPIDR2_EL0 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TPIDR2_EL0;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
        UNDEFINED;
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.nTPIDR2_EL0 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TPIDR2_EL0;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
        UNDEFINED;
    elseif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TPIDR2_EL0;
elseif PSTATE.EL == EL3 then
    X[t, 64] = TPIDR2_EL0;

```

MSR TPIDR2_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.EnTP2 == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.EnTP2 == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.nTPIDR2_EL0 == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TPIDR2_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
            UNDEFINED;
        elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.nTPIDR2_EL0 == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TPIDR2_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.EnTP2 == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.EnTP2 == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TPIDR2_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        TPIDR2_EL0 = X[t, 64];

```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TPIDR_EL0, EL0 Read/Write Software Thread ID Register

The TPIDR_EL0 characteristics are:

Purpose

Provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

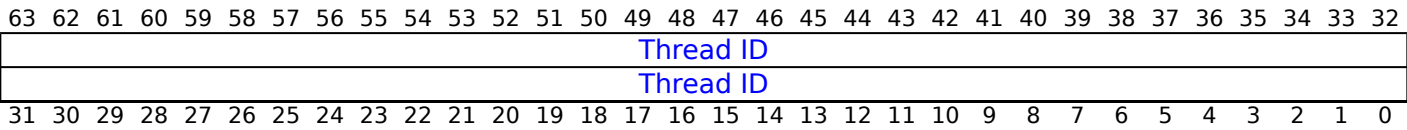
Configuration

AArch64 System register TPIDR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRURW\[31:0\]](#).

Attributes

TPIDR_EL0 is a 64-bit register.

Field descriptions



Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TPIDR_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TPIDR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b010

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.TPIDR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TPIDR_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.TPIDR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TPIDR_EL0;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TPIDR_EL0;
elseif PSTATE.EL == EL3 then
    X[t, 64] = TPIDR_EL0;

```

MSR TPIDR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b010

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.TPIDR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TPIDR_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGWTR_EL2.TPIDR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TPIDR_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    TPIDR_EL0 = X[t, 64];
elseif PSTATE.EL == EL3 then
    TPIDR_EL0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TPIDR_EL1, EL1 Software Thread ID Register

The TPIDR_EL1 characteristics are:

Purpose

Provides a location where software executing at EL1 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

Configuration

AArch64 System register TPIDR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRPRW\[31:0\]](#).

Attributes

TPIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Thread ID																															
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TPIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TPIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
    HFGTR_EL2.TPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TPIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TPIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TPIDR_EL1;

```

MSR TPIDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
    HFGWTR_EL2.TPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TPIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    TPIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TPIDR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TPIDRRO_EL0, EL0 Read-Only Software Thread ID Register

The TPIDRRO_EL0 characteristics are:

Purpose

Provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

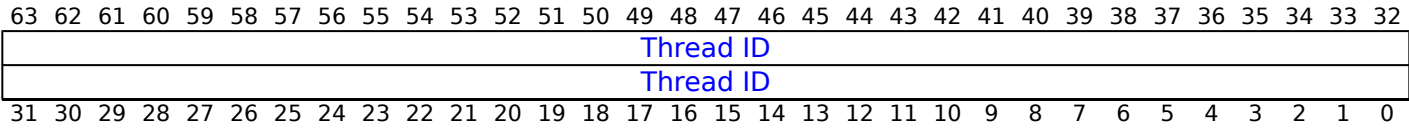
Configuration

AArch64 System register TPIDRRO_EL0 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRURO\[31:0\]](#).

Attributes

TPIDRRO_EL0 is a 64-bit register.

Field descriptions



Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

Accessing TPIDRRO_EL0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TPIDRRO_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && IsFeatureImplemented(FEAT_FGT) &&
(!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.TPIDRR0_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TPIDRR0_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.TPIDRR0_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TPIDRR0_EL0;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TPIDRR0_EL0;
elseif PSTATE.EL == EL3 then
    X[t, 64] = TPIDRR0_EL0;

```

MSR TPIDRR0_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGWTR_EL2.TPIDRR0_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TPIDRR0_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    TPIDRR0_EL0 = X[t, 64];
elseif PSTATE.EL == EL3 then
    TPIDRR0_EL0 = X[t, 64];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRBBASER_EL1 characteristics are:

Purpose

Defines the base address for the trace buffer.

Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBBASER_EL1 are UNDEFINED.

Attributes

TRBBASER_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																BASE															
BASE																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BASE, bits [63:12]

Trace Buffer Base pointer address. (TRBBASER_EL1.BASE < 12) is the address of the first byte in the trace buffer. Bits [11:0] of the Base pointer address are always zero. If the smallest implemented translation granule is not 4KB, then TRBBASER_EL1[N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value $\text{Log}_2(\text{smallest implemented translation granule})$.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [11:0]

Reserved, RES0.

Accessing TRBBASER_EL1

The PE might ignore a direct write to `TRBBASER_EL1` if `TRBLIMITR_EL1.E == 1`.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBBASER_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBBASER_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBBASER_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBBASER_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBBASER_EL1;

```

MSR TRBBASER_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBBASER_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBBASER_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBBASER_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBBASER_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRBIDR_EL1, Trace Buffer ID Register

The TRBIDR_EL1 characteristics are:

Purpose

Describes constraints on using the Trace Buffer Unit to software, including whether the Trace Buffer Unit can be programmed at the current Exception level.

Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBIDR_EL1 are UNDEFINED.

Attributes

TRBIDR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																EA				RES0		F	P	Align							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:12]

Reserved, RES0.

EA, bits [11:8] From Armv9.3:

External Abort handling. Describes how the PE manages External aborts on writes made by the Trace Buffer Unit to the trace buffer.

EA	Meaning
0b0000	Not described.
0b0001	The PE ignores External aborts on writes made by the Trace Buffer Unit.
0b0010	The External abort generates an SError interrupt at the PE.

All other values are reserved.

From Armv9.3, the value 0b0000 is not permitted.

[TRBIDR_EL1](#).EA describes only External aborts generated by the write to memory. External aborts on a translation table walk made by the Trace Buffer Unit generate trace buffer management events reported as MMU faults using [TRBSR_EL1](#).

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [7:6]

Reserved, RES0.

F, bit [5]

Flag updates. Describes how address translations performed by the Trace Buffer Unit manage the Access flag and dirty state.

F	Meaning
0b0	Hardware management of the Access flag and dirty state for accesses made by the Trace Buffer Unit is always disabled for all translation stages.
0b1	Hardware management of the Access flag and dirty state for accesses made by the Trace Buffer Unit is controlled in the same way as explicit memory accesses in the trace buffer owning translation regime.

Note

If hardware management of the Access flag is disabled for a stage of translation, an access to a Page or Block with the Access flag bit not set in the descriptor will generate an Access Flag fault.

If hardware management of the dirty state is disabled for a stage of translation, an access to a Page or Block will ignore the Dirty Bit Modifier in the descriptor and might generate a Permission fault, depending on the values of the access permission bits in the descriptor.

From Armv9.3, the value 0 is not permitted.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

P, bit [4]

Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the trace buffer is owned by a higher Exception level or another Security state. Defined values are:

P	Meaning
0b0	Programming is allowed.
0b1	Programming not allowed.

The value read from this field depends on the current Exception level and the Effective values of [MDCR_EL3.NSTB](#), [MDCR_EL3.NSTBE](#), and [MDCR_EL2.E2TB](#):

- If EL3 is implemented, and the owning Security state is Secure state, this field reads as one from:
 - Non-secure EL1 and Non-secure EL2.
 - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
 - If Secure EL2 is implemented and enabled, and [MDCR_EL2.E2TB](#) is 0b00, Secure EL1.
- If EL3 is implemented, and the owning Security state is Non-secure state, this field reads as one from:
 - Secure EL1.
 - If Secure EL2 is implemented, Secure EL2.
 - If EL2 is implemented and [MDCR_EL2.E2TB](#) is 0b00, Non-secure EL1.
 - If FEAT_RME is implemented, Realm EL1 and Realm EL2.
- If FEAT_RME is implemented, and the owning Security state is Realm state, this field reads as one from:
 - Non-secure EL1 and Non-secure EL2.
 - Secure EL1 and Secure EL2.
 - If [MDCR_EL2.E2TB](#) is 0b00, Realm EL1.

- If EL3 is not implemented, EL2 is implemented, and [MDCR_EL2.E2TB](#) is 0b00, this field reads as one from EL1.
- Otherwise, this field reads as zero.

Align, bits [3:0]

Defines the minimum alignment constraint for writes to [TRBPTR_EL1](#) and [TRBTRG_EL1](#). Defined values are:

Align	Meaning
0b0000	Byte.
0b0001	Halfword.
0b0010	Word.
0b0011	Doubleword.
0b0100	16 bytes.
0b0101	32 bytes.
0b0110	64 bytes.
0b0111	128 bytes.
0b1000	256 bytes.
0b1001	512 bytes.
0b1010	1KB.
0b1011	2KB.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.TRBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRBIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TRBIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBIDR_EL1;

```

3005/0907/2022 15:17:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRBLIMITR_EL1, Trace Buffer Limit Address Register

The TRBLIMITR_EL1 characteristics are:

Purpose

Defines the top address for the trace buffer, and controls the trace buffer modes and enable.

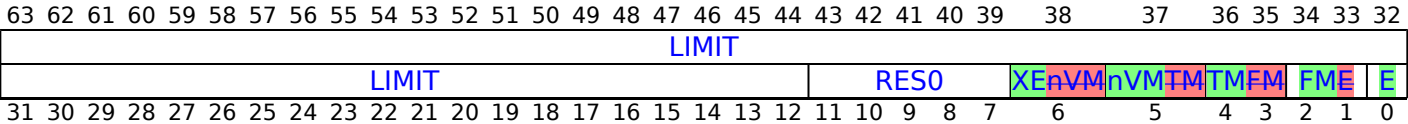
Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBLIMITR_EL1 are UNDEFINED.

Attributes

TRBLIMITR_EL1 is a 64-bit register.

Field descriptions



LIMIT, bits [63:12]

Trace Buffer Limit pointer address. (TRBLIMITR_EL1.LIMIT << 12) is the address of the last byte in the trace buffer plus one. Bits [11:0] of the Limit pointer address are always zero. If the smallest implemented translation granule is not 4KB, then TRBLIMITR_EL1[N-1:12] are TRBLIMITR_EL1.LIMIT << 12) is the address of the last byte in the trace buffer plus one. Bits [11:0] of the Limit pointer address are always zero. If the smallest implemented translation granule is not 4KB, then TRBLIMITR_EL1[N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value Log2(smallest implemented translation granule).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [11:7]

Reserved, RES0.

XE, bit [6] When FEAT_TRBE_EXT is implemented:

Trace Buffer Unit External mode enable. Used for save/restore of TRBLIMITR_EL1.XE.

XE	Meaning
0b0	Trace Buffer Unit is not enabled by this control.
0b1	If SelfHostedTraceEnabled() is FALSE, the Trace Buffer Unit is enabled.

Software must treat this field as UNK/SBZP when the OS Lock is unlocked.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Accessing this field has the following behavior:

- When !OSLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

nVM, bit [5]

Address mode.

nVM	Meaning
0b0	The trace buffer pointers are virtual addresses.
0b1	The trace buffer pointers are: <ul style="list-style-type: none"> Physical address in the owning security state if the owning translation regime has no stage 2 translation. Intermediate physical addresses in the owning security state if the owning translation regime has stage 2 translations.

When FEAT_TRBE_EXT is implemented and SelfHostedTraceEnabled() == FALSE, the trace buffer pointers are always physical addresses.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- AccessOn is a Cold reset, this field resets to an architecturally **RES1UNKNOWN** if all of the following are true:
 - FEAT_TRBE_EXT is implemented
 - !SelfHostedTraceEnabled()
- Otherwise, access to this field is **RW**.

TM, bits [4:3]

Trigger mode.

TM	Meaning
0b00	Stop on trigger. Flush then stop collection and raise maintenance interrupt on Trigger Event.
0b01	IRQ on trigger. Continue collection and raise maintenance interrupt on Trigger Event.
0b11	Ignore trigger. Continue collection and do not raise maintenance interrupt on Trigger Event.

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

FM, bits [2:1]

Trace buffer mode.

FM	Meaning
0b00	Fill mode. Stop collection and raise maintenance interrupt on current write pointer wrap.
0b01	Wrap mode. Continue collection and raise maintenance interrupt on current write pointer wrap.
0b11	Circular Buffer mode. Continue collection and do not raise maintenance interrupt on current write pointer wrap.

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Trace Buffer Unit enable. Controls whether the Trace Buffer Unit is enabled when SelfHostedTraceEnabled() == TRUE.

E	Meaning
0b0	Trace Buffer Unit is not enabled by this control. disabled.
0b1	If SelfHostedTraceEnabled() is TRUE, the Trace Buffer Unit is enabled enabled by this control.

If Regardless of the value of this field, the Trace Buffer Unit is disabled when SelfHostedTraceEnabled() == FALSE. All output is discarded by the Trace Buffer Unit when it is disabled. FEAT_TRBE_EXT is implemented and SelfHostedTraceEnabled() == FALSE, then TRBLIMITR_EL1.XE controls whether the Trace Buffer Unit is enabled.

If FEAT_TRBE_EXT is not implemented, then the Trace Buffer Unit is disabled when SelfHostedTraceEnabled() == FALSE.

All output is discarded by the Trace Buffer Unit when the Trace Buffer Unit is disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRBLIMITR_EL1

The PE might ignore a direct write to TRBLIMITR_EL1, other than a direct write that modifies TRBLIMITR_EL1.E, if TRBLIMITR_EL1.E == 1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBLIMITR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBLIMITR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBLIMITR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBLIMITR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBLIMITR_EL1;

```

MSR TRBLIMITR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBLIMITR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBLIMITR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBLIMITR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBLIMITR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRBMAR_EL1, Trace Buffer Memory Attribute Register

The TRBMAR_EL1 characteristics are:

Purpose

Controls Trace Buffer Unit accesses to memory.

If the trace buffer pointers specify virtual addresses, the address properties are defined by the translation tables and this register is ignored.

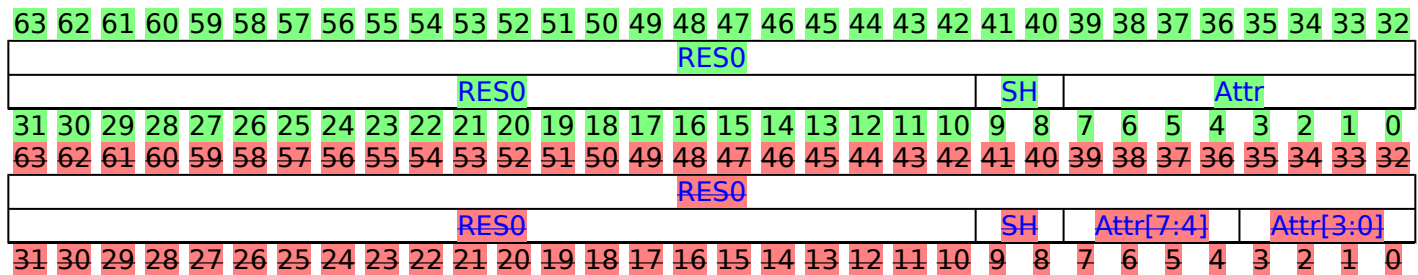
Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBMAR_EL1 are UNDEFINED.

Attributes

TRBMAR_EL1 is a 64-bit register.

Field descriptions



Bits [63:10]

Reserved, RES0.

SH, bits [9:8]

Trace buffer shareability domain. Defines the shareability domain for Normal memory used by the trace buffer.

SH	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

All other values are reserved.

This field is ignored when [TRBMAR_EL1.Attr](#) specifies any of the following memory types:

- Any Device memory type.
- Normal memory, Inner Non-cacheable, Outer Non-cacheable.

All Device and Normal Inner Non-cacheable Outer Non-cacheable memory regions are always treated as Outer Shareable.

The reset behavior of this field is:

- On a ColdWarm reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

Attr, [7:4], bits [7:04]

When TRBMAR_EL1.Attr[3:0] == 0bxxxx00000b0000:

Trace buffer memory type and attributes. Defines the memory type and, for Normal memory, the cacheability attributes, for memory addressed by the trace buffer.

Attr[7:4]	Meaning	Applies when
0x000b0000	Device-nGnRnE memory.	
0x400b0100	Normal memory, Inner Non-cacheable, Outer Non-cacheable with the XS attribute set to 0.	When FEAT_XS is implemented
0xA00b1010	Normal memory, Inner Write-through Cacheable, Outer Write-through Cacheable, Non-transient, Read-Allocate with the XS attribute set to 0.	When FEAT_XS is implemented
0xF00b1111	Tagged Normal memory, Outer Write-Back Non-transient, Read-allocate Write-allocate.	When FEAT_MTE2 is implemented

All other values are reserved.

The reset behavior of this field is:

- On a ColdWarm reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

When TRBMAR_EL1.Attr == 0b0000xxxx and TRBMAR_EL1.Attr != 0b00000000:

Trace buffer memory attributes. Defines the Device memory attributes for memory addressed by the trace buffer.

Attr	Meaning	Applies when
0x04	Device-nGnRE memory.	
0x08	Device-nGRE memory.	
0x0C	Device-GRE memory.	
0x01	Device-nGnRnE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x05	Device-nGnRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x09	Device-nGRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x0D	Device-GRE memory with the XS attribute set to 0.	When FEAT_XS is implemented

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

When TRBMAR_EL1.Attr != 0bxxxx0000 and TRBMAR_EL1.Attr != 0b0000xxxx:

Trace buffer memory type and attributes. Defines the memory type and, for Normal memory, the Outer and Inner cacheability attributes, for memory addressed by the trace buffer.

Attr	Meaning
0b0001xxxx	Normal memory, Outer Write-Through Transient, Write-allocate.
0b0010xxxx	Normal memory, Outer Write-Through Transient, Read-allocate.
0b0011xxxx	Normal memory, Outer Write-Through Transient, Read-allocate Write-allocate.
0b0100xxxx	Normal memory, Outer Non-cacheable.
0b0101xxxx	Normal memory, Outer Write-Back Transient, Write-allocate.
0b0110xxxx	Normal memory, Outer Write-Back Transient, Read-allocate.
0b0111xxxx	Normal memory, Outer Write-Back Transient, Read-allocate Write-allocate.
0b1000xxxx	Normal memory, Outer Write-Through Non-transient, No allocate.
0b1001xxxx	Normal memory, Outer Write-Through Non-transient, Write-allocate.
0b1010xxxx	Normal memory, Outer Write-Through Non-transient, Read-allocate.
0b1011xxxx	Normal memory, Outer Write-Through Non-transient, Read-allocate Write-allocate.
0b1100xxxx	Normal memory, Outer Write-Back Non-transient, No allocate.
0b1101xxxx	Normal memory, Outer Write-Back Non-transient, Write-allocate.
0b1110xxxx	Normal memory, Outer Write-Back Non-transient, Read-allocate.
0b1111xxxx	Normal memory, Outer Write-Back Non-transient, Read-allocate Write-allocate.
0bxxxx0001	Normal memory, Inner Write-Through Transient, Write-allocate.
0bxxxx0010	Normal memory, Inner Write-Through Transient, Read-allocate.
0bxxxx0011	Normal memory, Inner Write-Through Transient, Read-allocate Write-allocate.
0bxxxx0100	Normal memory, Inner Non-cacheable.
0bxxxx0101	Normal memory, Inner Write-Back Transient, Write-allocate.
0bxxxx0110	Normal memory, Inner Write-Back Transient, Read-allocate.
0bxxxx0111	Normal memory, Inner Write-Back Transient, Read-allocate Write-allocate.
0bxxxx1000	Normal memory, Inner Write-Through Non-transient, No allocate.
0bxxxx1001	Normal memory, Inner Write-Through Non-transient, Write-allocate.
0bxxxx1010	Normal memory, Inner Write-Through Non-transient, Read-allocate.
0bxxxx1011	Normal memory, Inner Write-Through Non-transient, Read-allocate Write-allocate.
0bxxxx1100	Normal memory, Inner Write-Back Non-transient, No allocate.
0bxxxx1101	Normal memory, Inner Write-Back Non-transient, Write-allocate.
0bxxxx1110	Normal memory, Inner Write-Back Non-transient, Read-allocate.
0bxxxx1111	Normal memory, Inner Write-Back Non-transient, Read-allocate Write-allocate.

The reset behavior of this field is:

- On a Cold reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

Otherwise:

Attr[7:4]	Meaning
0b0000	Device memory. The Device memory type is defined by TRBMAR_EL1.Attr[3:0].
0b0001	Normal memory, Outer Write-Through Transient, Write-allocate.
0b0010	Normal memory, Outer Write-Through Transient, Read-allocate.
0b0011	Normal memory, Outer Write-Through Transient, Read-allocate Write-allocate.
0b0100	Normal memory, Outer Non-cacheable.
0b0101	Normal memory, Outer Write-Back Transient, Write-allocate.
0b0110	Normal memory, Outer Write-Back Transient, Read-allocate.
0b0111	Normal memory, Outer Write-Back Transient, Read-allocate Write-allocate.
0b1000	Normal memory, Outer Write-Through Non-transient, No allocate.
0b1001	Normal memory, Outer Write-Through Non-transient, Write-allocate.
0b1010	Normal memory, Outer Write-Through Non-transient, Read-allocate.
0b1011	Normal memory, Outer Write-Through Non-transient, Read-allocate Write-allocate.
0b1100	Normal memory, Outer Write-Back Non-transient, No allocate.
0b1101	Normal memory, Outer Write-Back Non-transient, Write-allocate.
0b1110	Normal memory, Outer Write-Back Non-transient, Read-allocate.
0b1111	Normal memory, Outer Write-Back Non-transient, Read-allocate Write-allocate.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Reserved Trace buffer memory type and attributes. Defines the memory type and, for Normal memory, the Outer cacheability attributes, for memory addressed by the trace buffer. RES0.

Attr[3:0], bits [3:0]

When TRBMAR_EL1.Attr[7:4] == 0b0000:

Trace buffer memory attributes. Defines the Device memory attributes for memory addressed by the trace buffer.

Attr[3:0]	Meaning	Applies when
0b0000	Device-nGnRnE memory.	
0b0100	Device-nGnRE memory.	
0b1000	Device-nGRE memory.	
0b1100	Device-GRE memory.	
0b0001	Device-nGnRnE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0b0101	Device-nGnRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0b1001	Device-nGRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0b1101	Device-GRE memory with the XS attribute set to 0.	When FEAT_XS is implemented

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Trace buffer memory attributes. Defines the Inner cacheability attributes for memory addressed by the trace buffer.

Attr[3:0]	Meaning	Applies when
0b0000	The memory type is defined by TRBMAR_EL1.Attr[7:4].	When FEAT_MTE2 is implemented or FEAT_XS is implemented
0b0001	Normal memory, Inner Write-Through Transient, Write-allocate.	
0b0010	Normal memory, Inner Write-Through Transient, Read-allocate.	
0b0011	Normal memory, Inner Write-Through Transient, Read-allocate Write-allocate.	
0b0100	Normal memory, Inner Non-cacheable.	
0b0101	Normal memory, Inner Write-Back Transient, Write-allocate.	
0b0110	Normal memory, Inner Write-Back Transient, Read-allocate.	
0b0111	Normal memory, Inner Write-Back Transient, Read-allocate Write-allocate.	
0b1000	Normal memory, Inner Write-Through Non-transient, No allocate.	
0b1001	Normal memory, Inner Write-Through Non-transient, Write-allocate.	
0b1010	Normal memory, Inner Write-Through Non-transient, Read-allocate.	
0b1011	Normal memory, Inner Write-Through Non-transient, Read-allocate Write-allocate.	
0b1100	Normal memory, Inner Write-Back Non-transient, No allocate.	
0b1101	Normal memory, Inner Write-Back Non-transient, Write-allocate.	
0b1110	Normal memory, Inner Write-Back Non-transient, Read-allocate.	
0b1111	Normal memory, Inner Write-Back Non-transient, Read-allocate Write-allocate.	

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TRBMAR_EL1

The PE might ignore a direct write to TRBMAR_EL1 if `TRBLIMITR_EL1.E == 1`.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBMAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRBMAR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBMAR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBMAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBMAR_EL1;

```

MSR TRBMAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDBGWTR_EL2.TRBMAR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBMAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBMAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBMAR_EL1 = X[t, 64];

```

Accessing TRBMAR_EL1

The PE might ignore a write to [TRBMAR_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1.E](#) == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1.XE](#) == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBMAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBMAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRBMAR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRBMAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBMAR_EL1;

```

MSR TRBMAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBMAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBMAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBMAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBMAR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBPTR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBPTR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBPTR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBPTR_EL1;

```

MSR TRBPTR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBPTR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBPTR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBPTR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBPTR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRBSR_EL1, Trace Buffer Status/syndrome Register

The TRBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software for a trace buffer management event.

Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBSR_EL1 are UNDEFINED.

Attributes

TRBSR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
RES0								MSS2																						
EC	RES0	DAT	IRQ	IRQ	TRG	TRG	WRAP	WRAP	RES0	RES0	EA	EA	SRES0	RES0	MSS															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Bits [63:56:32]

Reserved, RES0.

MSS2, bits [55:32]

Management event Specific Syndrome 2. Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

MSS2 encoding for other trace buffer management events

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0											

Bits [23:0]

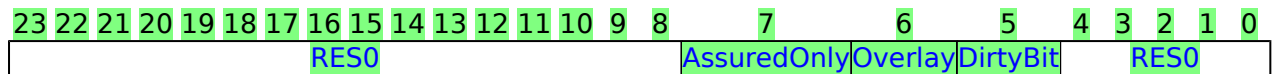
Reserved, RES0.

MSS2 encoding for a buffer management event for an IMPLEMENTATION DEFINED reason

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												IMPLEMENTATION DEFINED											

IMPLEMENTATION DEFINED, bits [23:0]

IMPLEMENTATION DEFINED.

MSS2 encoding for stage 1 or stage 2 Data Aborts on write to trace buffer**Bits [23:8]**

Reserved, RES0.

AssuredOnly, bit [7]

When FEAT_THE is implemented, TRBSR_EL1.EC == 0b100101 and GetTRBSR_EL1_FSC() == 0b0011xx:

AssuredOnly flag. If a memory access generates a stage 2 Data Abort, then this field holds information about the fault.

AssuredOnly	Meaning
0b0	Data Abort is not due to AssuredOnly.
0b1	Data Abort is due to AssuredOnly.

Otherwise:

Reserved, RES0.

Overlay, bit [6]

When (FEAT_S1POE is implemented or FEAT_S2POE is implemented) and GetTRBSR_EL1_FSC() == 0b0011xx:

Overlay flag. If a memory access generates a Data Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Data Abort due to Base Permissions.
0b1	Data Abort due to Overlay Permissions.

Otherwise:

Reserved, RES0.

DirtyBit, bit [5]

When (FEAT_S1PIE is implemented or FEAT_S2PIE is implemented) and GetTRBSR_EL1_FSC() == 0b0011xx:

DirtyBit flag. If a memory access generates a Data Abort (Write Access) for a Permission fault (When using Indirect Permission), then this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to state of nDirty / Dirty bit.
0b1	Permission Fault is due to state of nDirty / Dirty bit.

Otherwise:

Reserved, RES0.

Bits [4:0]

Reserved, RES0.

EC, bits [31:26]

Event class. Top-level description of the cause of the trace buffer management event.

EC	Meaning	MSS	Applies when
0b000000	Other trace buffer management event. All trace buffer management events other than those described by the other defined Event class codes.	MSS encoding for Granule Protection Check faultMSS encoding for other trace buffer management events	
0b011110	Granule Protection Check fault on write to trace buffer, other than Granule Protection Fault (GPF). That on is, write any to of trace the following: buffer. <ul style="list-style-type: none"> Granule Protection Table (GPT) address size fault. GPT walk fault. Synchronous External abort on GPT fetch. A GPF on translation table walk or update is reported as either a Stage 1 or Stage 2 Data Abort, as appropriate. Other GPFs are reported as a Stage 1 Data Abort.	MSS encoding for other trace buffer management eventsMSS encoding for Granule Protection Check fault	When FEAT_RME is implemented
0b011111	Buffer management event for an IMPLEMENTATION DEFINED reason.	MSS encoding for Buffer management event for IMPLEMENTATION DEFINED reason	
0b100100	Stage 1 Data Abort on write to trace buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer	
0b100101	Stage 2 Data Abort on write to trace buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer	

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [25:2423]

Reserved, RES0.

DAT, bit [23]**When FEAT_TRBE_EXT is implemented:**

Data. Indicates when the Trace Buffer Unit has trace data that has not yet been written to memory.

DAT	Meaning
0b0	Internal buffers are empty. All Trace operations Accepted by the Trace Buffer Unit will Complete in finite time.
0b1	Internal buffers are not empty.

When [TRBSR_EL1](#).{DAT, S} is {0, 1}, meaning Collection is stopped and the Trace Buffer Unit internal buffers are empty, then all trace data has been written to memory. An additional Data Synchronization Barrier may be required to ensure that the writes are Complete. When [TRBSR_EL1](#).DAT is 0 and Collection is not stopped, there may still be trace data held by the trace unit that the Trace Buffer Unit has not Accepted.

That is, [TRBSR_EL1](#).DAT reads as 1 when the Trace Buffer Unit has Accepted trace data from the trace unit, but has not yet written it to memory.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

IRQ, bit [22]

Maintenance interrupt status.

IRQ	Meaning
0b0	Maintenance interrupt is not asserted.
0b1	Maintenance interrupt is asserted.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

TRG, bit [21]

Triggered.

TRG	Meaning
0b0	No Detected Trigger has been observed since this field was last cleared to zero.
0b1	A Detected Trigger has been observed since this field was last cleared to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

WRAP, bit [20]

Wrapped.

WRAP	Meaning
0b0	The current write pointer has not wrapped since this field was last cleared to zero.
0b1	The current write pointer has wrapped since this field was last cleared to zero.

For each byte of trace the Trace Buffer Unit Accepts and writes to the trace buffer at the address in the current write pointer, if the current write pointer is equal to the Limit pointer minus one, the current write pointer is wrapped by setting it to the Base pointer, and this field is set to 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bit [19]

Reserved, RES0.

EA, bit [18]

When the PE sets this bit as the result of an External Abort:

External Abort.

EA	Meaning
0b0	An External Abort has not been asserted.
0b1	An External Abort has been asserted and detected by the Trace Buffer Unit.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When the PE never sets this field as the result of an External Abort, access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

S, bit [17]

Stopped.

S	Meaning
0b0	Collection has not been stopped.
0b1	Collection is stopped.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bit [16]

Reserved, RES0.

MSS, bits [15:0]

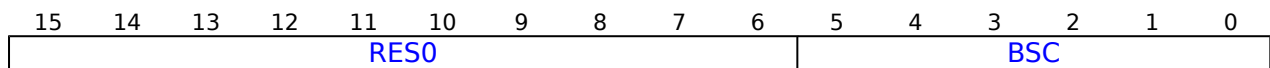
Management Event Specific Syndrome. Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

MSS encoding for other trace buffer management events



Bits [15:6]

Reserved, RES0.

BSC, bits [5:0]

Trace buffer status code.

BSC	Meaning	Applies when
0b000000	Collection not stopped, or access not allowed.	
0b000001	Trace buffer filled. Collection stopped because the current write pointer wrapped to the base pointer and the trace buffer mode is Fill mode.	
0b000010	Trigger Event. Collection stopped because of a Trigger Event. See TRBTRG_EL1 for more information.	
0b000011	Manual Stop. Collection stopped because of a Manual Stop event. See TRBCR.ManStop for more information.	When FEAT_TRBE_EXT is implemented

All other values are reserved.

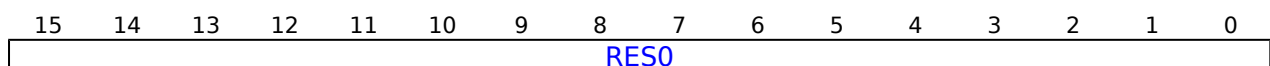
MSS encoding for Buffer management event for IMPLEMENTATION DEFINED reason



IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED.

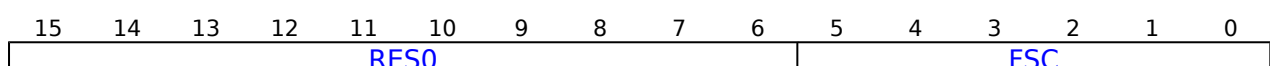
MSS encoding for Granule Protection Check fault



Bits [15:0]

Reserved, RES0.

MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer



Bits [15:6]

Reserved, RES0.

FSC, bits [5:0]

Fault status code.

FSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Asynchronous External abort.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented

0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

Accessing TRBSR_EL1

The PE might ignore a direct write to [TRBSR_EL1](#) if [TRBLIMITR_EL1](#).E == 1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBSR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRBSR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBSR_EL1;

```

MSR TRBSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBSR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRBSR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBSR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRBTRG_EL1 characteristics are:

Specifies the number of bytes of trace to capture following a Detected Trigger before a Trigger Event.

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBTRG_EL1 are UNDEFINED.

TRBTRG_EL1 is a 64-bit register.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																
																TRG																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Reserved, RES0.

Trigger count.

Specifies the number of bytes of trace to capture following a Detected Trigger before a Trigger Event.

TRBTRG_EL1 decrements by 1 for every byte of trace written to the trace buffer when all of the following are true:

- TRBTRG_EL1 is nonzero.
- TRBSR_EL1.TRG is 1.

The architecture places restrictions on the values that software can write to the counter.

As a result of the restrictions an implementation might treat some of TRG[M:0] as RES0, where M is defined by [TRBIDR_EL1.Align](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

The PE might ignore a direct write to TRBTRG_EL1 if [TRBLIMITR_EL1.E](#) == 1.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRBTRG_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRBTRG_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRBTRG_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRBTRG_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRBTRG_EL1;
    
```

MSR TRBTRG_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1011	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRBTRG_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.E2TB == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBTRG_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSTB[0] == '0' || MDCR_EL3.NSTB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSTBE != SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRBTRG_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRBTRG_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL1, bit [17]**When TRCIDR6.EXLEVEL_RL_EL1 == 1:**

Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state.

EXLEVEL_RL_EL1	Meaning
0b0	When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1.
0b1	When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL0, bit [16]**When TRCIDR6.EXLEVEL_RL_EL0 == 1:**

Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state.

EXLEVEL_RL_EL0	Meaning
0b0	When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0.
0b1	When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [15]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [14]**When Non-secure EL2 is implemented:**

Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state.

EXLEVEL_NS_EL2	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL2.
0b1	The Address Comparator does not perform comparisons in Non-secure EL2.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL1, bit [13]**When Non-secure EL1 is implemented:**

Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state.

EXLEVEL_NS_EL1	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL1.
0b1	The Address Comparator does not perform comparisons in Non-secure EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL0, bit [12]**When Non-secure EL0 is implemented:**

Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state.

EXLEVEL_NS_EL0	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL0.
0b1	The Address Comparator does not perform comparisons in Non-secure EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL3, bit [11]**When EL3 is implemented:**

EL3 address comparison control. Controls whether a comparison can occur at EL3.

EXLEVEL_S_EL3	Meaning
0b0	The Address Comparator performs comparisons in EL3.
0b1	The Address Comparator does not perform comparisons in EL3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL2, bit [10]**When EL2 is implemented and FEAT_SEL2 is implemented:**

Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state.

EXLEVEL_S_EL2	Meaning
0b0	The Address Comparator performs comparisons in Secure EL2.
0b1	The Address Comparator does not perform comparisons in Secure EL2.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL1, bit [9]**When Secure EL1 is implemented:**

Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state.

EXLEVEL_S_EL1	Meaning
0b0	The Address Comparator performs comparisons in Secure EL1.
0b1	The Address Comparator does not perform comparisons in Secure EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_ELO, bit [8]**When Secure EL0 is implemented:**

Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state.

EXLEVEL_S_ELO	Meaning
0b0	The Address Comparator performs comparisons in Secure EL0.
0b1	The Address Comparator does not perform comparisons in Secure EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [7]

Reserved, RES0.

CONTEXT, bits [6:4]**When TRCIDR4.NUMCIDC != 0b0000 or TRCIDR4.NUMVMIDC != 0b0000:**

Selects a Context Identifier Comparator or Virtual Context Identifier Comparator:

CONTEXT	Meaning	Applies when
0b000	Comparator 0.	
0b001	Comparator 1.	When UInt(TRCIDR4.NUMCIDC) > 1 or UInt(TRCIDR4.NUMVMIDC) > 1
0b010	Comparator 2.	When UInt(TRCIDR4.NUMCIDC) > 2 or UInt(TRCIDR4.NUMVMIDC) > 2
0b011	Comparator 3.	When UInt(TRCIDR4.NUMCIDC) > 3 or UInt(TRCIDR4.NUMVMIDC) > 3
0b100	Comparator 4.	When UInt(TRCIDR4.NUMCIDC) > 4 or UInt(TRCIDR4.NUMVMIDC) > 4
0b101	Comparator 5.	When UInt(TRCIDR4.NUMCIDC) > 5 or UInt(TRCIDR4.NUMVMIDC) > 5
0b110	Comparator 6.	When UInt(TRCIDR4.NUMCIDC) > 6 or UInt(TRCIDR4.NUMVMIDC) > 6
0b111	Comparator 7.	When UInt(TRCIDR4.NUMCIDC) > 7 or UInt(TRCIDR4.NUMVMIDC) > 7

The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CONTEXTTYPE, bits [3:2]**When TRCIDR4.NUMCIDC != 0b0000 or TRCIDR4.NUMVMIDC != 0b0000:**

Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons.

CONTEXTTYPE	Meaning	Applies when
0b00	The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators.	
0b01	The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match.	When TRCIDR4.NUMCIDC != 0b0000
0b10	The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match.	When TRCIDR4.NUMVMIDC != 0b0000
0b11	The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match.	When TRCIDR4.NUMCIDC != 0b0000 and TRCIDR4.NUMVMIDC != 0b0000

If [TRCIDR4.NUMCIDC](#) == 0b0000, then bit [2] is RES0.

If [TRCIDR4.NUMVMIDC](#) == 0b0000, then bit [3] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [1:0]

Reserved, RES0.

Accessing TRCACATR<n>

Must be programmed if any of the following are true:

- [TRCBBCTL](#)R.RANGE[n/2] == 1.
- [TRCRSCTL](#)R<a>.GROUP == 0b0100 and [TRCRSCTL](#)R<a>.SAC[n] == 1.
- [TRCRSCTL](#)R<a>.GROUP == 0b0101 and [TRCRSCTL](#)R<a>.ARC[n/2] == 1.
- [TRCVIIECTL](#)R.EXCLUDE[n/2] == 1.
- [TRCVIIECTL](#)R.INCLUDE[n/2] == 1.
- [TRCVISSCTL](#)R.START[n] == 1.
- [TRCVISSCTL](#)R.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- [TRCQCTL](#)R.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCACATR<m> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:0b0	0b01:m[3]

```

integer m = UInt(op2<0>:CRm<3:1>);

if m >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCACATR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCACATR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[m];

```

MSR TRCACATR<m>, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:0b0	0b01:m[3]

```

integer m = UInt(op2<0>:CRm<3:1>);

if m >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elseif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[m] = X[t, 64];
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCACVR<n>, Address Comparator Value Register <n>, n = 0 - 15

The TRCACVR<n> characteristics are:

Purpose

Contains the address value.

Configuration

AArch64 System register TRCACVR<n> bits [63:0] are architecturally mapped to External register [TRCACVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $\text{UInt}(\text{TRCIDR4.NUMACPAIRS}) * 2 > n$. Otherwise, direct accesses to TRCACVR<n> are UNDEFINED.

Attributes

TRCACVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ADDRESS															
																ADDRESS															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Address Value.

The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>.

The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an UNKNOWN value, where P is defined as the maximum virtual address size supported by the PE.

The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCACVR<n>

Must be programmed if any of the following are true:

- [TRCBBCTLR](#).RANGE[n/2] == 1.
- [TRCRSCTLR<a>](#).GROUP == 0b0100 and [TRCRSCTLR<a>](#).SAC[n] == 1.
- [TRCRSCTLR<a>](#).GROUP == 0b0101 and [TRCRSCTLR<a>](#).ARC[n/2] == 1.

- [TRCVIIECTLR.EXCLUDE\[n/2\]](#) == 1.
- [TRCVIIECTLR.INCLUDE\[n/2\]](#) == 1.
- [TRCVISSCTLR.START\[n\]](#) == 1.
- [TRCVISSCTLR.STOP\[n\]](#) == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- [TRCQCTL.RANGE\[n/2\]](#) == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCACVR<m> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:0b0	0b00:m[3]

```
integer m = UInt(op2<0>:CRm<3:1>);

if m >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCACVR[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCACVR[m];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCACVR[m];
```

MSR TRCACVR<m>, <Xt> ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:0b0	0b00:m[3]

```

integer m = UInt(op2<0>:CRm<3:1>);

if m >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCACVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCACVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCACVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Configuration

AArch64 System register TRCAUTHSTATUS bits [31:0] are architecturally mapped to External register [TRCAUTHSTATUS\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCAUTHSTATUS are UNDEFINED.

Attributes

TRCAUTHSTATUS is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0				RTNID		RTID		RES0								RLNID		RLID		HNID		HID		SNID		SID		NSNID		NSID	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

RTID	Meaning
0b00	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLNID.

RLID, bits [13:12]

Realm invasive debug.

RLID	Meaning
0b00	Not implemented.

HNID, bits [11:10]

Hyp Non-invasive Debug. Indicates whether a separate enable control for EL2 non-invasive debug features is implemented and enabled.

HNID	Meaning
0b00	Separate Hyp non-invasive debug enable not implemented, or EL2 non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

HID, bits [9:8]

Hyp Invasive Debug. Indicates whether a separate enable control for EL2 invasive debug features is implemented and enabled.

HID	Meaning
0b00	Separate Hyp invasive debug enable not implemented, or EL2 invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

SNID, bits [7:6]

Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled.

SNID	Meaning
0b00	Secure non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b00.

When EL3 is not implemented and the PE is Secure, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

SID, bits [5:4]

Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled.

SID	Meaning
0b00	Secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

NSNID, bits [3:2]

Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled.

NSNID	Meaning
0b00	Non-secure non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field reads as 0b11.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b11.

When EL3 is not implemented and the PE is Secure, this field reads as 0b00.

NSID, bits [1:0]

Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled.

NSID	Meaning
0b00	Non-secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

Accessing TRCAUTHSTATUS

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCAUTHSTATUS

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1110	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCAUTHSTATUS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCAUTHSTATUS;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCAUTHSTATUS;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCAUTHSTATUS;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCAUXCTLR, Auxiliary Control Register

The TRCAUXCTLR characteristics are:

Purpose

The function of this register is IMPLEMENTATION DEFINED.

Configuration

AArch64 System register TRCAUXCTLR bits [31:0] are architecturally mapped to External register [TRCAUXCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCAUXCTLR are UNDEFINED.

Attributes

TRCAUXCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

This field reads as an IMPLEMENTATION DEFINED value and writes to this field have IMPLEMENTATION DEFINED behavior.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Accessing TRCAUXCTLR

If this register is nonzero then it might cause the behavior of a trace unit to contradict this architecture specification. See the documentation of the specific implementation for information about the IMPLEMENTATION DEFINED support for this register.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCAUXCTLR

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b10	0b001	0b0000	0b0110	0b000
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
& HDFGRTR_EL2.TRCAUXCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCAUXCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCAUXCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCAUXCTLR;

```

MSR TRCAUXCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCAUXCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCAUXCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCAUXCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCBBCTLR characteristics are:

Purpose

Controls the regions in the memory map where branch broadcasting is active.

Configuration

AArch64 System register TRCBBCTLR bits [31:0] are architecturally mapped to External register [TRCBBCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, TRCIDR0.TRCBB == 1 and UInt(TRCIDR4.NUMACPAIRS) > 0. Otherwise, direct accesses to TRCBBCTLR are UNDEFINED.

Attributes

TRCBBCTLR is a 64-bit register.

Field descriptions

6362616059585756555453525150494847464544434241	40	39	38	37	36	35	34
RES0							
RES0			MODE	RANGE[7]	RANGE[6]	RANGE[5]	RANGE[4]
31302928272625242322212019181716151413121110			9	8	7	6	5
						4	3
							2

Bits [63:9]

Reserved, RES0.

MODE, bit [8]

Mode.

MODE	Meaning
0b0	Exclude Mode. Branch broadcasting is not active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then branch broadcasting is active for all instructions.
0b1	Include Mode. Branch broadcasting is active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then the behavior of the trace unit is CONSTRAINED UNPREDICTABLE. That is, the trace unit might or might not consider any instructions to be in a branch broadcasting region.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RANGE[<m>], bit [m], for m = 7 to 0

Address range field.

Selects whether Address Range Comparator <m> is used with branch broadcasting.

RANGE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected.
0b1	The address range that Address Range Comparator <m> defines, is selected.

This bit is RES0 if m >= [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCBBCTLR

Must be programmed if [TRCCONFIGR](#).BB == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCBBCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCBBCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCBBCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCBBCTLR;

```

MSR TRCBBCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCBBCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCBBCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCBBCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCCCTLR, Cycle Count Control Register

The TRCCCCTLR characteristics are:

Purpose

Set the threshold value for cycle counting.

Configuration

AArch64 System register TRCCCCTLR bits [31:0] are architecturally mapped to External register [TRCCCCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, [FEAT_TRC_SR is implemented](#) and TRCIDR0.TRCCCI == 1. Otherwise, direct accesses to TRCCCCTLR are UNDEFINED.

Attributes

TRCCCCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
												RES0																			
RES0												THRESHOLD																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:12]

Reserved, RES0.

THRESHOLD, bits [11:0]

Sets the threshold value for instruction trace cycle counting.

The minimum threshold value that can be programmed into THRESHOLD is given in [TRCIDR3.CCITMIN](#). If the THRESHOLD value is smaller than the value in [TRCIDR3.CCITMIN](#) then the behavior is CONSTRAINED UNPREDICTABLE. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known.

Writing a value of zero when [TRCCONFIGR.CCI](#) enables instruction trace cycle counting results in CONSTRAINED UNPREDICTABLE behavior. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCCCTLR

Must be programmed if [TRCCONFIGR.CCI](#) == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCCCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCCCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCCCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCCCTLR;

```

MSR TRCCCCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCCCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCCCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCCCTLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCCTLR0, Context Identifier Comparator Control Register 0

The TRCCIDCCTLR0 characteristics are:

Purpose

Contains Context identifier mask values for the [TRCCIDCVR<n>](#) registers, for n = 0 to 3.

Configuration

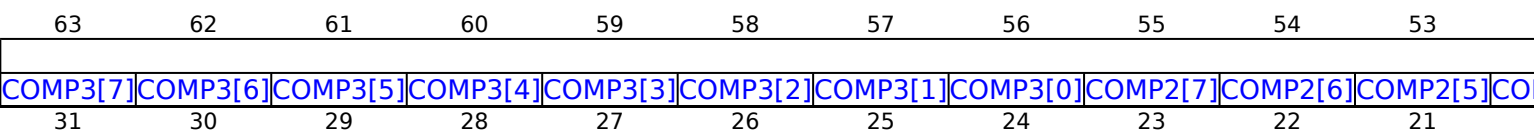
AArch64 System register TRCCIDCCTLR0 bits [31:0] are architecturally mapped to External register [TRCCIDCCTLR0\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 0 \times 0$ and $\text{UInt}(\text{TRCIDR2.CIDSIZE}) > 0$. Otherwise, direct accesses to TRCCIDCCTLR0 are UNDEFINED.

Attributes

TRCCIDCCTLR0 is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, RES0.

COMP3[<m>], bit [m+24], for m = 7 to 0 When UInt(TRCIDR4.NUMCIDC) > 3:

TRCCIDCVR3 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR3. Each bit in this field corresponds to a byte in TRCCIDCVR3.

COMP3[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR3[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR3[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP2[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 2:

TRCCIDCVR2 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR2. Each bit in this field corresponds to a byte in TRCCIDCVR2.

COMP2[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR2[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR2[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP1[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 1:

TRCCIDCVR1 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR1. Each bit in this field corresponds to a byte in TRCCIDCVR1.

COMP1[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR1[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR1[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP0[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 0:

TRCCIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0.

COMP0[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2](#).CIDSIZE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCCIDCCTLR0

If software uses the [TRCCIDCVR<n>](#) registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCCIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCCIDCVR<n>](#) is not 0x00, the behavior of the Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCIDCCTLR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCCTLR0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCCTLR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCCTLR0;

```

MSR TRCCIDCCTLR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCCTLR0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCCTLR0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCCTLR0 = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCCTLR1, Context Identifier Comparator Control Register 1

The TRCCIDCCTLR1 characteristics are:

Purpose

Contains Context identifier mask values for the [TRCCIDCVR<n>](#) registers, for n = 4 to 7.

Configuration

AArch64 System register TRCCIDCCTLR1 bits [31:0] are architecturally mapped to External register [TRCCIDCCTLR1\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 0x4$ and $\text{UInt}(\text{TRCIDR2.CIDSIZE}) > 0$. Otherwise, direct accesses to TRCCIDCCTLR1 are UNDEFINED.

Attributes

TRCCIDCCTLR1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53
COMP7[7] COMP7[6] COMP7[5] COMP7[4] COMP7[3] COMP7[2] COMP7[1] COMP7[0] COMP6[7] COMP6[6] COMP6[5] COMP6[4]										
31	30	29	28	27	26	25	24	23	22	21

Bits [63:32]

Reserved, RES0.

COMP7[<m>], bit [m+24], for m = 7 to 0 When UInt(TRCIDR4.NUMCIDC) > 7:

TRCCIDCVR7 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR7. Each bit in this field corresponds to a byte in TRCCIDCVR7.

COMP7[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR7[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR7[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP6[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 6:

TRCCIDCVR6 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR6. Each bit in this field corresponds to a byte in TRCCIDCVR6.

COMP6[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR6[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR6[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP5[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 5:

TRCCIDCVR5 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR5. Each bit in this field corresponds to a byte in TRCCIDCVR5.

COMP5[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR5[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR5[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP4[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 4:

TRCCIDCVR4 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR4. Each bit in this field corresponds to a byte in TRCCIDCVR4.

COMP4[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR4[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR4[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2](#).CIDSIZE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCCIDCCTLR1

If software uses the [TRCCIDCVR<n>](#) registers, for n = 4 to 7, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCCIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCCIDCVR<n>](#) is not 0x00, the behavior of the Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCIDCCTLR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCCTLR1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCCTLR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCCTLR1;

```

MSR TRCCIDCCTLR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCCTLR1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCCTLR1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCCTLR1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n>, n = 0 - 7

The TRCCIDCVR<n> characteristics are:

Purpose

Contains a Context identifier value.

Configuration

AArch64 System register TRCCIDCVR<n> bits [63:0] are architecturally mapped to External register [TRCCIDCVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $UInt(TRCIDR4.NUMCIDC) > n$. Otherwise, direct accesses to TRCCIDCVR<n> are UNDEFINED.

Attributes

TRCCIDCVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																VALUE															
																VALUE															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VALUE, bits [63:0]

Context identifier value. The width of this field is indicated by [TRCIDR2.CIDSIZE](#). Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCIDCVR<n>

Must be programmed if any of the following are true:

- [TRCRSCTLR<a>](#).GROUP == 0b0110 and [TRCRSCTLR<a>](#).CID[n] == 1.
- [TRCACATR<a>](#).CONTEXTTYPE == 0b01 or 0b11 and [TRCACATR<a>](#).CONTEXT == n.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCIDCVR<m> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:0b0	0b000

```

integer m = UInt(CRm<3:1>);

if m >= NUM_TRACE_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCVR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCIDCVR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[m];

```

MSR TRCCIDCVR<m>, <Xt> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:0b0	0b000

```

integer m = UInt(CRm<3:1>);

if m >= NUM_TRACE_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCIDCVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCCLAIMCLR, Claim Tag Clear Register

The TRCCLAIMCLR characteristics are:

Purpose

In conjunction with [TRCCLAIMSET](#), provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configuration

AArch64 System register TRCCLAIMCLR bits [31:0] are architecturally mapped to External register [TRCCLAIMCLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCCLAIMCLR are UNDEFINED.

Attributes

TRCCLAIMCLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50
CLR[31]	CLR[30]	CLR[29]	CLR[28]	CLR[27]	CLR[26]	CLR[25]	CLR[24]	CLR[23]	CLR[22]	CLR[21]	CLR[20]	CLR[19]	CLR[18]
31	30	29	28	27	26	25	24	23	22	21	20	19	18

Bits [63:32]

Reserved, RES0.

CLR[<m>], bit [m], for m = 31 to 0

Claim Tag Clear. Indicates the current status of Claim Tag bit <m>, and is used to clear Claim Tag bit <m> to 0.

CLR[<m>]	Meaning
0b0	On a read: Claim Tag bit <m> is not set. On a write: Ignored.
0b1	On a read: Claim Tag bit <m> is set. On a write: Clear Claim tag bit <m> to 0.

The number of Claim Tag bits implemented is indicated in [TRCCLAIMSET](#).

This bit reads-as-zero and ignores writes if m > the number of Claim Tag bits.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Access to this field is **W1C**.

Accessing TRCCLAIMCLR

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCLAIMCLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCLAIMCLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCLAIMCLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCLAIMCLR;

```

MSR TRCCLAIMCLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCLAIMCLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCLAIMCLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCLAIMCLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCCLAIMSET, Claim Tag Set Register

The TRCCLAIMSET characteristics are:

Purpose

In conjunction with [TRCCLAIMCLR](#), provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configuration

AArch64 System register TRCCLAIMSET bits [31:0] are architecturally mapped to External register [TRCCLAIMSET\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCCLAIMSET are UNDEFINED.

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

Attributes

TRCCLAIMSET is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	
SET[31]	SET[30]	SET[29]	SET[28]	SET[27]	SET[26]	SET[25]	SET[24]	SET[23]	SET[22]	SET[21]	SET[20]	SET[19]	SET[18]	SET[17]
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17

Bits [63:32]

Reserved, RES0.

SET[<m>], bit [m], for m = 31 to 0

Claim Tag Set. Indicates whether Claim Tag bit <m> is implemented, and is used to set Claim Tag bit <m> to 1.

SET[<m>]	Meaning
0b0	On a read: Claim Tag bit <m> is not implemented. On a write: Ignored.
0b1	On a read: Claim Tag bit <m> is implemented. On a write: Set Claim Tag bit <m> to 1.

This bit reads-as-zero and ignores writes if m > the number of Claim Tag bits.

Access to this field is **RAO/W1S**.

Accessing TRCCLAIMSET

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCLAIMSET

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCLAIMSET;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCLAIMSET;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCLAIMSET;

```

MSR TRCCLAIMSET, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCLAIM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCLAIMSET = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCLAIMSET = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCNTCTLR<n>, Counter Control Register <n>, n = 0 - 3

The TRCCNTCTLR<n> characteristics are:

Purpose

Controls the operation of Counter <n>.

Configuration

AArch64 System register TRCCNTCTLR<n> bits [31:0] are architecturally mapped to External register [TRCCNTCTLR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and `UInt(TRCIDR5.NUMCNTR) > n`. Otherwise, direct accesses to TRCCNTCTLR<n> are UNDEFINED.

Attributes

TRCCNTCTLR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35			
RES0																															
RES0															CNTCHAIN	RLDSELF	RLDEVENT_TYPE				RES0	RLDEVENT_SEL				CNTEVENT_TYPE				RES0	CNTEVENT
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3			

Bits [63:18]

Reserved, RES0.

CNTCHAIN, bit [17]

For TRCCNTCTLR3 and TRCCNTCTLR1, this field controls whether the Counter decrements when a reload event occurs for Counter <n-1>.

CNTCHAIN	Meaning
0b0	The Counter does not decrement when a reload event for Counter <n-1> occurs.
0b1	Counter <n> decrements when a reload event for Counter <n-1> occurs. This concatenates Counter <n> and Counter <n-1>, to provide a larger count value.

CNTCHAIN is not implemented for TRCCNTCTLR0 and TRCCNTCTLR2.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RLDSELF, bit [16]

Controls whether a reload event occurs for the Counter, when the Counter reaches zero.

RLDSELF	Meaning
0b0	Normal mode. The Counter is in Normal mode.
0b1	Self-reload mode. The Counter is in Self-reload mode.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RLDEVENT_TYPE, bit [15]

Chooses the type of Resource Selector.

Selects an event, that when it occurs causes a reload event for Counter <n>.

RLDEVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [14:13]

Reserved, RES0.

RLDEVENT_SEL, bits [12:8]

Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Selects an event, that when it occurs causes a reload event for Counter <n>.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

CNTEVENT_TYPE, bit [7]

Chooses the type of Resource Selector.

Selects an event, that when it occurs causes Counter <n> to decrement.

CNTEVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

CNTEVENT_SEL, bits [4:0]

Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Selects an event, that when it occurs causes Counter <n> to decrement.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCNTCTLR<n>

Must be programmed if [TRCRSCTLR<a>.GROUP == 0b0010](#) and [TRCRSCTLR<a>.COUNTERS\[n\] == 1](#).

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCNTCTLR<m> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b01:m[1:0]	0b101

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTCTLR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTCTLR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[m];

```

MSR TRCCNTCTLR<m>, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b01:m[1:0]	0b101

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTCTLR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTCTLR[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCNTRLDVR<n>, Counter Reload Value Register <n>, n = 0 - 3

The TRCCNTRLDVR<n> characteristics are:

Purpose

This sets or returns the reload count value for Counter <n>.

Configuration

AArch64 System register TRCCNTRLDVR<n> bits [31:0] are architecturally mapped to External register [TRCCNTRLDVR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $UInt(TRCIDR5.NUMCNTR) > n$. Otherwise, direct accesses to TRCCNTRLDVR<n> are UNDEFINED.

Attributes

TRCCNTRLDVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																RES0																					
RES0																VALUE																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

Bits [63:16]

Reserved, RES0.

VALUE, bits [15:0]

Contains the reload value for Counter <n>. When a reload event occurs for Counter <n> then the trace unit copies the VALUE<n> field into Counter <n>.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCNTRLDVR<n>

Must be programmed if [TRCRSCTLR<a>](#).GROUP == 0b0010 and [TRCRSCTLR<a>](#).COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCNTRLDVR<m> ; Where m = 0-3

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b10	0b001	0b0000	0b00:m[1:0]	0b101
------	-------	--------	-------------	-------

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTRLDVR[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = TRCCNTRLDVR[m];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTRLDVR[m];

```

MSR TRCCNTRLDVR<m>, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b00:m[1:0]	0b101

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTRLDVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTRLDVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTRLDVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCNTVR<n>, Counter Value Register <n>, n = 0 - 3

The TRCCNTVR<n> characteristics are:

Purpose

This sets or returns the value of Counter <n>.

Configuration

AArch64 System register TRCCNTVR<n> bits [31:0] are architecturally mapped to External register [TRCCNTVR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and UInt(TRCIDR5.NUMCNTR) > n. Otherwise, direct accesses to TRCCNTVR<n> are UNDEFINED.

Attributes

TRCCNTVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																VALUE															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

VALUE, bits [15:0]

Contains the count value of Counter.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCNTVR<n>

Must be programmed if [TRCRSCTLR<a>.GROUP == 0b0010](#) and [TRCRSCTLR<a>.COUNTERS\[n\] == 1](#).

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCNTVR<m> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b10:m[1:0]	0b101

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCCNTVRn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTVR[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTVR[m];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCNTVR[m];

```

MSR TRCCNTVR<m>, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b10:m[1:0]	0b101


```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCCNTVRn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCNTVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCONFIGR, Trace Configuration Register

The TRCCONFIGR characteristics are:

Purpose

Controls the tracing options.

Configuration

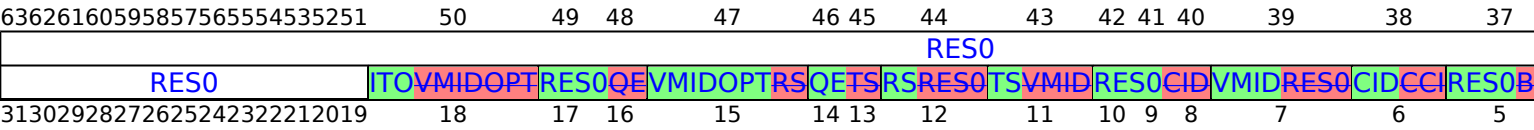
AArch64 System register TRCCONFIGR bits [31:0] are architecturally mapped to External register [TRCCONFIGR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCCONFIGR are UNDEFINED.

Attributes

TRCCONFIGR is a 64-bit register.

Field descriptions



Bits [63:19]16

Reserved, RES0.

ITO, bit [18]
When TRCIDR0.ITE == 1:

Instrumentation Trace Override.

ITO	Meaning
0b0	Instrumentation Trace Override disabled.
0b1	Instrumentation Trace Override enabled.

This field is ignored when SelfHostedTraceEnabled() returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [17:16]

Reserved, RES0.

VMIDOPT, bit [15]**When TRCIDR2.VMIDOPT == 0b01:**

Virtual context identifier selection control.

VMIDOPT	Meaning
0b0	VTTBR_EL2 .VMID is used as the Virtual context identifier.
0b1	CONTEXTIDR_EL2 .PROCID is used as the Virtual context identifier.

When TRCIDR2.VMIDOPT == 0b00:

Reserved, RES0.

Virtual context identifier selection control.

[VTTBR_EL2](#).VMID is used as the Virtual context identifier.

When TRCIDR2.VMIDOPT == 0b10:

Reserved, RES1.

Virtual context identifier selection control.

[CONTEXTIDR_EL2](#).PROCID is used as the Virtual context identifier.

Otherwise:

Reserved, RES0.

QE, bits [14:13]**When TRCIDR0.QSUPP == 0b01:**

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b01	Q elements with instruction counts are enabled.
	Q elements without instruction counts are disabled.

All other values are reserved.

When TRCIDR0.QSUPP == 0b10:

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b11	Q elements with instruction counts are enabled.
	Q elements without instruction counts are enabled.

All other values are reserved.

When TRCIDR0.QSUPP == 0b11:

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b01	Q elements with instruction counts are enabled.
	Q elements without instruction counts are disabled.
0b11	Q elements with instruction counts are enabled.
	Q elements without instruction counts are enabled.

All other values are reserved.

Otherwise:

Reserved, RES0.

RS, bit [12]**When TRCIDR0.RETSTACK == 1:**

Return stack control.

RS	Meaning
0b0	Return stack is disabled.
0b1	Return stack is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TS, bit [11]**When TRCIDR0.TSSIZE != 0b00000:**

Global timestamp tracing control.

TS	Meaning
0b0	Global timestamp tracing is disabled.
0b1	Global timestamp tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [10:8]

Reserved, RES0.

VMID, bit [7]**When TRCIDR2.VMIDSIZE != 0b000000:**

Virtual context identifier tracing control.

VMID	Meaning
0b0	Virtual context identifier tracing is disabled.
0b1	Virtual context identifier tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CID, bit [6]**When TRCIDR2.CIDSIZE != 0b000000:**

Context identifier tracing control.

CID	Meaning
0b0	Context identifier tracing is disabled.
0b1	Context identifier tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [5]

Reserved, RES0.

CCI, bit [4]**When TRCIDR0.TRCCCI == 1:**

Cycle counting instruction tracing control.

CCI	Meaning
0b0	Cycle counting instruction tracing is disabled.
0b1	Cycle counting instruction tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BB, bit [3]
When TRCIDR0.TRCBB == 1:

Branch broadcasting control.

BB	Meaning
0b0	Branch broadcasting is disabled.
0b1	Branch broadcasting is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [2:1]

Reserved, RES0.

Bit [0]

Reserved, RES1.

Accessing TRCCONFIGR

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCCONFIGR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCONFIGR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCCONFIGR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCONFIGR;

```

MSR TRCCONFIGR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCONFIGR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCCONFIGR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCONFIGR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCDEVARCH, Device Architecture Register

The TRCDEVARCH characteristics are:

Purpose

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

AArch64 System register TRCDEVARCH bits [31:0] are architecturally mapped to External register [TRCDEVARCH\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCDEVARCH are UNDEFINED.

Attributes

TRCDEVARCH is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

ARCHITECT, bits [31:21]

Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.

ARCHITECT	Meaning
0b01000111011	JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.

Other values are defined by the JEDEC JEP106 standard.

This field reads as 0x23B.

PRESENT, bit [20]

DEVARCH Present. Defines that the DEVARCH register is present.

PRESENT	Meaning
0b0	Device Architecture information not present.
0b1	Device Architecture information present.

This field reads as 1.

REVISION, bits [19:16]

Revision. Defines the architecture revision of the component.

REVISION	Meaning
0b0000	ETEv1.0, FEAT_ETE.
0b0001	ETEv1.1, FEAT_ETEv1p1.
0b0010	ETEv1.2, FEAT_ETEv1p2.
0b0011	ETEv1.3, FEAT_ETEv1p3.

All other values are reserved.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component.

ARCHVER	Meaning
0b0101	ETEv1.

ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12].

This field reads as 0x5.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

ARCHPART	Meaning
0xA13	Arm PE trace architecture.

ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0].

This field reads as 0xA13.

Accessing TRCDEVARCH

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCDEVARCH

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1111	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCDEVARCH;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCDEVARCH;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCDEVARCH;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCDEVID characteristics are:

Purpose

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

AArch64 System register TRCDEVID bits [31:0] are architecturally mapped to External register [TRCDEVID\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCDEVID are UNDEFINED.

Attributes

TRCDEVID is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																
																RES0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:0]

Reserved, RES0.

Accessing TRCDEVID

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCDEVID

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b0010	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCDEVID;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCDEVID;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCDEVID;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCEVENTCTL0R, Event Control 0 Register

The TRCEVENTCTL0R characteristics are:

Purpose

Controls the generation of ETEEvents.

Configuration

AArch64 System register TRCEVENTCTL0R bits [31:0] are architecturally mapped to External register [TRCEVENTCTL0R\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC SR is implemented** and TRCIDR4.NUMRSPAIR != 0b0000. Otherwise, direct accesses to TRCEVENTCTL0R are UNDEFINED.

Attributes

TRCEVENTCTL0R is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	
RES0																											
EVENT3_TYPE	RES0	EVENT3_SEL	EVENT2_TYPE	RES0	EVENT2_SEL	EVENT1_TYPE	RES0	EVENT1_SEL	EVENT0_TYPE	RES0																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	

Bits [63:32]

Reserved, RES0.

EVENT3_TYPE, bit [31]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 3:

Chooses the type of Resource Selector.

EVENT3_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT3.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT3.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [30:29]

Reserved, RES0.

EVENT3_SEL, bits [28:24]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 3:

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.TYPE controls whether TRCEVENTCTL0R.EVENT3.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R.INSTEN\[3\]](#) == 1, then Event element 3 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT2_TYPE, bit [23]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 2:

Chooses the type of Resource Selector.

EVENT2_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT2.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT2.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [22:21]

Reserved, RES0.

EVENT2_SEL, bits [20:16]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 2:

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.TYPE controls whether TRCEVENTCTL0R.EVENT2.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R.INSTEN\[2\]](#) == 1, then Event element 2 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT1_TYPE, bit [15]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 1:

Chooses the type of Resource Selector.

EVENT1_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT1.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT1.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [14:13]

Reserved, RES0.

EVENT1_SEL, bits [12:8]**When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 1:**

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.TYPE controls whether TRCEVENTCTL0R.EVENT1.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R.INSTEN\[1\]](#) == 1, then Event element 1 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT0_TYPE, bit [7]**When TRCIDR4.NUMRSPAIR != 0b0000:**

Chooses the type of Resource Selector.

EVENT0_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT0.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT0.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [6:5]

Reserved, RES0.

EVENT0_SEL, bits [4:0]**When TRCIDR4.NUMRSPAIR != 0b0000:**

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.TYPE controls whether TRCEVENTCTL0R.EVENT0.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R](#).INSTEN[0] == 1, then Event element 0 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCEVENTCTL0R

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCEVENTCTL0R

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEVENTCTL0R;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEVENTCTL0R;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEVENTCTL0R;

```

MSR TRCEVENTCTL0R, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEVENTCTL0R = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEVENTCTL0R = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEVENTCTL0R = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCEVENTCTL1R, Event Control 1 Register

The TRCEVENTCTL1R characteristics are:

Purpose

Controls the behavior of the ETEEvents that [TRCEVENTCTL0R](#) selects.

Configuration

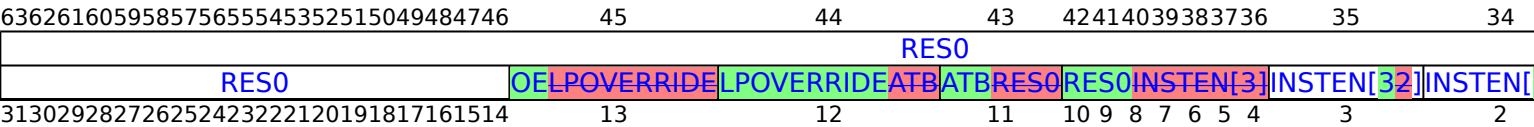
AArch64 System register TRCEVENTCTL1R bits [31:0] are architecturally mapped to External register [TRCEVENTCTL1R\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCEVENTCTL1R are UNDEFINED.

Attributes

TRCEVENTCTL1R is a 64-bit register.

Field descriptions



Bits [63:14]13

Reserved, RES0.

OE, bit [13]
When TRCIDR5.OE == 1:

ETE Trace Output Enable control.

OE	Meaning
0b0	Trace output to any IMPLEMENTATION DEFINED trace output interface is disabled.
0b1	Trace output to any IMPLEMENTATION DEFINED trace output interface is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Otherwise:

Reserved, RES0.

LPOVERRIDE, bit [12]**When TRCIDR5.LPOVERRIDE == 1:**

Low-power Override Mode select.

LPOVERRIDE	Meaning
0b0	Trace unit Low-power Override Mode is not enabled. That is, the trace unit is permitted to enter low-power state.
0b1	Trace unit Low-power Override Mode is enabled. That is, entry to a low-power state does not affect the trace unit resources or trace generation.

Otherwise:

Reserved, RES0.

ATB, bit [11]**When TRCIDR5.ATBTRIG == 1:**

AMBA Trace Bus (ATB) trigger enable.

If a CoreSight ATB interface is implemented then when ETEEvent 0 occurs the trace unit sets:

- ATID == 0x7D.
- ATDATA to the value of [TRCTRACEIDR](#).

If the width of ATDATA is greater than the width of [TRCTRACEIDR](#).TRACEID then the trace unit zeros the upper ATDATA bits.

If ETEEvent 0 is programmed to occur based on program execution, such as an Address Comparator, the ATB trigger might not be inserted into the ATB stream at the same time as any trace generated by that program execution is output by the trace unit. Typically, the generated trace might be buffered in a trace unit which means that the ATB trigger would be output before the associated trace is output.

If ETEEvent 0 is asserted multiple times in close succession, the trace unit is required to generate an ATB trigger for the first assertion, but might ignore one or more of the subsequent assertions. Arm recommends that the window in which ETEEvent 0 is ignored is limited only by the time taken to output an ATB trigger.

ATB	Meaning
0b0	ATB trigger is disabled.
0b1	ATB trigger is enabled.

Otherwise:

Reserved, RES0.

Bits [10:4]

Reserved, RES0.

INSTEN[<m>], bit [m], for m = 3 to 0

Event element control.

INSTEN[<m>]	Meaning
0b0	The trace unit does not generate an Event element <m>.
0b1	The trace unit generates an Event element <m>.

This bit is RES0 if m >= the number indicated by [TRCIDR0](#).NUMEVENT.

Accessing TRCEVENTCTL1R

Must be programmed.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCEVENTCTL1R

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEVENTCTL1R;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEVENTCTL1R;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEVENTCTL1R;

```

MSR TRCEVENTCTL1R, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEVENTCTL1R = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEVENTCTL1R = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEVENTCTL1R = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCEXTINSEL<n>, External Input Select Register <n>, n = 0 - 3

The TRCEXTINSEL<n> characteristics are:

Purpose

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSEL is an alias of TRCEXTINSEL0.

Configuration

AArch64 System register TRCEXTINSEL<n> bits [31:0] are architecturally mapped to External register [TRCEXTINSEL<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $UInt(TRCIDR5.NUMEXTINSEL) > n$. Otherwise, direct accesses to TRCEXTINSEL<n> are UNDEFINED.

Attributes

TRCEXTINSEL<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32					
																RES0																				
RES0																evtCount																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					

Bits [63:16]

Reserved, RES0.

evtCount, bits [15:0]

PMU event to select.

The event number as defined by the Arm ARM.

Software must program this field with a PMU event that is supported by the PE being programmed.

There are three ranges of PMU event numbers:

- PMU event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events.
- PMU event numbers in the range 0x0040 to 0x00BF are Arm recommended common architectural and microarchitectural PMU events.
- PMU event numbers in the range 0x00C0 to 0x03FF are IMPLEMENTATION DEFINED PMU events.

If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type:

- For the range 0x0000 to 0x003F, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field.

- For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

UNPREDICTABLE means the PMU event must not expose privileged information.

Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common IMPLEMENTATION DEFINED PMU events, then no PMU event is counted and the value read back on evtCount is the value written.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCEXTINSELR<n>

Must be programmed if any of the following is true: [TRCRSCTLR<a>](#).GROUP == 0b0000 and [TRCRSCTLR<a>](#).EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCEXTINSELR<m> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b10:m[1:0]	0b100

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEXTINSELN[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCEXTINSELN[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELN[m];

```

MSR TRCEXTINSELN<m>, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b10:m[1:0]	0b100

```

integer m = UInt(CRm<1:0>);

if m >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEXTINSELN[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEXTINSELN[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCEXTINSELN[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR0, ID Register 0

The TRCIDR0 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR0 bits [31:0] are architecturally mapped to External register [TRCIDR0\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR0 are UNDEFINED.

Attributes

TRCIDR0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46
RES0	COMMTRANS	COMMOPT	TSSIZE	TSMARK	ITERES0	RES0	TRCEXDATA	TRCEXDATA	QSUPP	QSUPP	QFILT	QFILT	CONL				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14

Bits [63:31]

Reserved, RES0.

COMMTRANS, bit [30]

Transaction Start element behavior.

COMMTRANS	Meaning
0b0	Transaction Start elements are P0 elements.
0b1	Transaction Start elements are not P0 elements.

COMMOPT, bit [29]

Indicates the contents and encodings of Cycle count packets.

COMMOPT	Meaning
0b0	Commit mode 0.
0b1	Commit mode 1.

The Commit mode defines the contents and encodings of Cycle Count packets, in particular how Commit elements are indicated by these packets. See the descriptions of these packets for more details.

Accessing this field has the following behavior:

- Access is **RAO/WI** if all of the following are true:
 - TRCIDR0.TRCCCI == 1
 - UInt(TRCIDR8.MAXSPEC) == 0x0
- When TRCIDR0.TRCCCI == 0, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

TSSIZE, bits [28:24]

Indicates that the trace unit implements Global timestamping and the size of the timestamp value.

TSSIZE	Meaning
0b00000	Global timestamping not implemented.
0b01000	Global timestamping implemented with a 64-bit timestamp value.

All other values are reserved.

This field reads as 0b01000.

TSMARK, bit [23]**When FEAT_ETEv1p1 is implemented:**

Indicates whether Timestamp Marker elements are generated.

TSMARK	Meaning
0b0	Timestamp Marker elements are not generated.
0b1	Timestamp Marker elements are generated.

Otherwise:

Reserved, RES0.

ITE, Bits bit [22:18]**When FEAT_ETEv1p3 is implemented:**

Indicates whether Instrumentation Trace is implemented.

ITE	Meaning
0b0	Instrumentation Trace not implemented.
0b1	Instrumentation Trace implemented.

This field has the value 1 if FEAT_ITE is implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [21:18]

Reserved, RES0.

TRCEXDATA, bit [17]**When TRCIDR0.TRCDATA != 0b00:**

Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

TRCEXDATA	Meaning
0b0	Tracing of data transfers for exceptions and exception returns not implemented.
0b1	Tracing of data transfers for exceptions and exception returns implemented.

Otherwise:

Reserved, RES0.

QSUPP, bits [16:15]

Indicates that the trace unit implements Q element support.

QSUPP	Meaning
0b00	Q element support is not implemented.
0b01	Q element support is implemented, and only supports Q elements with instruction counts.
0b10	Q element support is implemented, and only supports Q elements without instruction counts.
0b11	Q element support is implemented, and supports: <ul style="list-style-type: none"> • Q elements with instruction counts. • Q elements without instruction counts.

QFILT, bit [14]

Indicates if the trace unit implements Q element filtering.

QFILT	Meaning
0b0	Q element filtering is not implemented.
0b1	Q element filtering is implemented.

If TRCIDR0.QSUPP == 0b00 then this field is 0.

CONDTYPE, bits [13:12]**When TRCIDR0.TRCCOND == 1:**

Indicates how conditional instructions are traced. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

CONDTYPE	Meaning
0b00	Conditional instructions are traced with an indication of whether they pass or fail their condition code check.
0b01	Conditional instructions are traced with an indication of the APSR condition flags.

All other values are reserved.

Otherwise:

Reserved, RES0.

NUMEVENT, bits [11:10]**When TRCIDR4.NUMRSPAIR == 0b0000:**

Indicates the number of ETEEvents implemented.

NUMEVENT	Meaning
0b00	The trace unit supports 0 ETEEvents.

All other values are reserved.

When TRCIDR4.NUMRSPAIR != 0b0000:

Indicates the number of ETEEvents implemented.

NUMEVENT	Meaning
0b00	The trace unit supports 1 ETEEvent.
0b01	The trace unit supports 2 ETEEvents.
0b10	The trace unit supports 3 ETEEvents.
0b11	The trace unit supports 4 ETEEvents.

Otherwise:

Reserved, RES0.

RETSTACK, bit [9]

Indicates if the trace unit supports the return stack.

RETSTACK	Meaning
0b0	Return stack not implemented.
0b1	Return stack implemented.

Bit [8]

Reserved, RES0.

TRCCCI, bit [7]

Indicates if the trace unit implements cycle counting.

TRCCCI	Meaning
0b0	Cycle counting not implemented.
0b1	Cycle counting implemented.

This field reads as 1.

TRCCOND, bit [6]

Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures.

TRCCOND	Meaning
0b0	Conditional instruction tracing not implemented.
0b1	Conditional instruction tracing implemented.

This field reads as 0.

TRCBB, bit [5]

Indicates if the trace unit implements branch broadcasting.

TRCBB	Meaning
0b0	Branch broadcasting not implemented.
0b1	Branch broadcasting implemented.

This field reads as 1.

TRCDATA, bits [4:3]

Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures.

TRCDATA	Meaning
0b00	Data tracing not implemented.
0b11	Data tracing implemented.

All other values are reserved.

This field reads as 0b00.

INSTP0, bits [2:1]

Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures.

INSTP0	Meaning
0b00	Load and store instructions are not P0 instructions.
0b11	Load and store instructions are P0 instructions.

All other values are reserved.

This field reads as 0b00.

Bit [0]

Reserved, RES1.

Accessing TRCIDR0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR1, ID Register 1

The TRCIDR1 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR1 bits [31:0] are architecturally mapped to External register [TRCIDR1\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR1 are UNDEFINED.

Attributes

TRCIDR1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
DESIGNER								RES0								RES1				TRCARCHMAJ				TRCARCHMIN				REVISION			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

DESIGNER, bits [31:24]

Indicates which company designed the trace unit. The permitted values of this field are the same as [MIDR_EL1](#).Implementer.

Bits [23:16]

Reserved, RES0.

Bits [15:12]

Reserved, RES1.

TRCARCHMAJ, bits [11:8]

Major architecture version.

TRCARCHMAJ	Meaning
0b1111	If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to TRCDEVARCH .

All other values are reserved.

This field reads as 0b1111.

TRCARCHMIN, bits [7:4]

Minor architecture version.

TRCARCHMIN	Meaning
0b1111	If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to TRCDEVARCH .

All other values are reserved.

This field reads as 0b1111.

REVISION, bits [3:0]

Implementation revision.

Returns an IMPLEMENTATION DEFINED value that identifies the revision of the trace unit.

Arm deprecates any use of this field and recommends that implementations set this field to zero.

Accessing TRCIDR1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR10, ID Register 10

The TRCIDR10 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR10 bits [31:0] are architecturally mapped to External register [TRCIDR10\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR10 are UNDEFINED.

Attributes

TRCIDR10 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMP1KEY																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMP1KEY, bits [31:0]

When TRCIDR0.TRCDATA != 0b00:

Indicates the number of P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR10

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR10

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR11, ID Register 11

The TRCIDR11 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR11 bits [31:0] are architecturally mapped to External register [TRCIDR11\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR11 are UNDEFINED.

Attributes

TRCIDR11 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMP1SPC																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMP1SPC, bits [31:0]

When TRCIDR0.TRCDATA != 0b00:

Indicates the number of special P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR11

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b110


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR12, ID Register 12

The TRCIDR12 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR12 bits [31:0] are architecturally mapped to External register [TRCIDR12\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR12 are UNDEFINED.

Attributes

TRCIDR12 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMCONDKEY																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMCONDKEY, bits [31:0]

When TRCIDR0.TRCCOND == 1:

Indicates the number of conditional instruction right-hand keys. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR12

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR12

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR13, ID Register 13

The TRCIDR13 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR13 bits [31:0] are architecturally mapped to External register [TRCIDR13\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR13 are UNDEFINED.

Attributes

TRCIDR13 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMCONDSPC																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMCONDSPC, bits [31:0]

When TRCIDR0.TRCCOND == 1:

Indicates the number of special conditional instruction right-hand keys. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR13

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR13;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR2, ID Register 2

The TRCIDR2 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR2 bits [31:0] are architecturally mapped to External register [TRCIDR2\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR2 are UNDEFINED.

Attributes

TRCIDR2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
WFXMODE	VMIDOPT	CCSIZE	DVSIZE	DASIZE	VMIDSIZE	CIDSIZE	IASIZE																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

WFXMODE, bit [31]

Indicates whether WFI, WFIT, WFE, and WFET instructions are classified as P0 instructions:

WFXMODE	Meaning
0b0	WFI, WFIT, WFE, and WFET instructions are not classified as P0 instructions.
0b1	WFI, WFIT, WFE, and WFET instructions are classified as P0 instructions.

VMIDOPT, bits [30:29]

Indicates the options for Virtual context identifier selection.

VMIDOPT	Meaning
0b00	Virtual context identifier selection not supported. TRCCONFIGR.VMIDOPT is RES0.
0b01	Virtual context identifier selection supported. TRCCONFIGR.VMIDOPT is implemented.
0b10	Virtual context identifier selection not supported. TRCCONFIGR.VMIDOPT is RES1.

All other values are reserved.

If TRCIDR2.VMIDSIZE == 0b000000 then this field is 0b00.

If TRCIDR2.VMIDSIZE != 0b000000 then this field is 0b10.

CCSIZE, bits [28:25]

When TRCIDR0.TRCCCI == 1:

Indicates the size of the cycle counter.

CCSIZE	Meaning
0b0000	The cycle counter is 12 bits in length.
0b0001	The cycle counter is 13 bits in length.
0b0010	The cycle counter is 14 bits in length.
0b0011	The cycle counter is 15 bits in length.
0b0100	The cycle counter is 16 bits in length.
0b0101	The cycle counter is 17 bits in length.
0b0110	The cycle counter is 18 bits in length.
0b0111	The cycle counter is 19 bits in length.
0b1000	The cycle counter is 20 bits in length.

All other values are reserved.

Otherwise:

Reserved, RES0.

DVSIZE, bits [24:20]

When TRCIDR0.TRCDATA != 0b00:

Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

DVSIZE	Meaning
0b00000	Data value tracing not implemented.
0b00100	Data value tracing has a maximum of 32-bit data values.
0b01000	Data value tracing has a maximum of 64-bit data values.

All other values are reserved.

Otherwise:

Reserved, RES0.

DASIZE, bits [19:15]

When TRCIDR0.TRCDATA != 0b00:

Indicates the data address size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

DASIZE	Meaning
0b00000	Data address tracing not implemented.
0b00100	Data address tracing has a maximum of 32-bit data addresses.
0b01000	Data address tracing has a maximum of 64-bit data addresses.

All other values are reserved.

Otherwise:

Reserved, RES0.

VMIDSIZE, bits [14:10]

Indicates the trace unit Virtual context identifier size.

VMIDSIZE	Meaning
0b00000	Virtual context identifier tracing is not supported.
0b00001	8-bit Virtual context identifier size.
0b00010	16-bit Virtual context identifier size.
0b00100	32-bit Virtual context identifier size.

All other values are reserved.

If the PE does not implement EL2 then this field is 0b00000.

If the PE implements EL2 then this field is 0b00100.

CIDSIZE, bits [9:5]

Indicates the Context identifier size.

CIDSIZE	Meaning
0b00000	Context identifier tracing is not supported.
0b00100	32-bit Context identifier size.

All other values are reserved.

This field reads as 0b00100.

IASIZE, bits [4:0]

Virtual instruction address size.

IASIZE	Meaning
0b00100	Maximum of 32-bit instruction address size.
0b01000	Maximum of 64-bit instruction address size.

All other values are reserved.

This field reads as 0b01000.

Accessing TRCIDR2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR3, ID Register 3

The TRCIDR3 characteristics are:

Purpose

Returns the base architecture of the trace unit.

Configuration

AArch64 System register TRCIDR3 bits [31:0] are architecturally mapped to External register [TRCIDR3\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR3 are UNDEFINED.

Attributes

TRCIDR3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52
NOOVERFLOW	NUMPROC[2:0]	SYSSTALL	STALLCTL	SYNCPR	TRCERR	RES0	EXLEVEL_NS_EL2	EXLEVEL_NS_EL1	EXLEVEL		R
31	30	29	28	27	26	25	24	23	22	21	20

Bits [63:32]

Reserved, RES0.

NOOVERFLOW, bit [31]

Indicates if overflow prevention is implemented.

NOOVERFLOW	Meaning
0b0	Overflow prevention is not implemented.
0b1	Overflow prevention is implemented.

If TRCIDR3.STALLCTL == 0 then this field is 0.

NUMPROC, bits [13:12, 30:28]

Indicates the number of PEs available for tracing.

NUMPROC	Meaning
0b00000	The trace unit can trace one PE.

This field reads as 0b00000.

The NUMPROC field is split as follows:

- NUMPROC[2:0] is TRCIDR3[30:28].
- NUMPROC[4:3] is TRCIDR3[13:12].

SYSSTALL, bit [27]

Indicates if stalling of the PE is permitted.

SYSSTALL	Meaning
0b0	Stalling of the PE is not permitted.
0b1	Stalling of the PE is permitted.

The value of this field might be dynamic and change based on system conditions.

If TRCIDR3.STALLCTL == 0 then this field is 0.

STALLCTL, bit [26]

Indicates if trace unit implements stalling of the PE.

STALLCTL	Meaning
0b0	Stalling of the PE is not implemented.
0b1	Stalling of the PE is implemented.

SYNCPR, bit [25]

Indicates if an implementation has a fixed synchronization period.

SYNCPR	Meaning
0b0	TRCSYNCPR is read/write so software can change the synchronization period.
0b1	TRCSYNCPR is read-only so the synchronization period is fixed.

This field reads as 0.

TRCERR, bit [24]

Indicates forced tracing of System Error exceptions is implemented.

TRCERR	Meaning
0b0	Forced tracing of System Error exceptions is not implemented.
0b1	Forced tracing of System Error exceptions is implemented.

This field reads as 1.

Bit [23]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [22]

Indicates if Non-secure EL2 is implemented.

EXLEVEL_NS_EL2	Meaning
0b0	Non-secure EL2 is not implemented.
0b1	Non-secure EL2 is implemented.

EXLEVEL_NS_EL1, bit [21]

Indicates if Non-secure EL1 is implemented.

EXLEVEL_NS_EL1	Meaning
0b0	Non-secure EL1 is not implemented.
0b1	Non-secure EL1 is implemented.

EXLEVEL_NS_EL0, bit [20]

Indicates if Non-secure EL0 is implemented.

EXLEVEL_NS_EL0	Meaning
0b0	Non-secure EL0 is not implemented.
0b1	Non-secure EL0 is implemented.

EXLEVEL_S_EL3, bit [19]

Indicates if EL3 is implemented.

EXLEVEL_S_EL3	Meaning
0b0	EL3 is not implemented.
0b1	EL3 is implemented.

EXLEVEL_S_EL2, bit [18]

Indicates if Secure EL2 is implemented.

EXLEVEL_S_EL2	Meaning
0b0	Secure EL2 is not implemented.
0b1	Secure EL2 is implemented.

EXLEVEL_S_EL1, bit [17]

Indicates if Secure EL1 is implemented.

EXLEVEL_S_EL1	Meaning
0b0	Secure EL1 is not implemented.
0b1	Secure EL1 is implemented.

EXLEVEL_S_EL0, bit [16]

Indicates if Secure EL0 is implemented.

EXLEVEL_S_EL0	Meaning
0b0	Secure EL0 is not implemented.
0b1	Secure EL0 is implemented.

Bits [15:14]

Reserved, RES0.

CCITMIN, bits [11:0]

Indicates the minimum value that can be programmed in [TRCCCCTLR.THRESHOLD](#).

If [TRCIDR0.TRCCCI](#) == 1 then the minimum value of this field is 0x001.

If [TRCIDR0.TRCCCI](#) == 0 then this field is zero.

Accessing TRCIDR3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR4, ID Register 4

The TRCIDR4 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR4 bits [31:0] are architecturally mapped to External register [TRCIDR4\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR4 are UNDEFINED.

Attributes

TRCIDR4 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMVMIDC								NUMCIDC								NUMSSCC								NUMRSPAIR							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMVMIDC, bits [31:28]

Indicates the number of Virtual Context Identifier Comparators that are available for tracing.

NUMVMIDC	Meaning
0b0000	No Virtual Context Identifier Comparators are available.
0b0001	The implementation has one Virtual Context Identifier Comparator.
0b0010	The implementation has two Virtual Context Identifier Comparators.
0b0011	The implementation has three Virtual Context Identifier Comparators.
0b0100	The implementation has four Virtual Context Identifier Comparators.
0b0101	The implementation has five Virtual Context Identifier Comparators.
0b0110	The implementation has six Virtual Context Identifier Comparators.
0b0111	The implementation has seven Virtual Context Identifier Comparators.
0b1000	The implementation has eight Virtual Context Identifier Comparators.

All other values are reserved.

NUMCIDC, bits [27:24]

Indicates the number of Context Identifier Comparators that are available for tracing.

NUMCIDC	Meaning
0b0000	No Context Identifier Comparators are available.
0b0001	The implementation has one Context Identifier Comparator.
0b0010	The implementation has two Context Identifier Comparators.
0b0011	The implementation has three Context Identifier Comparators.
0b0100	The implementation has four Context Identifier Comparators.
0b0101	The implementation has five Context Identifier Comparators.
0b0110	The implementation has six Context Identifier Comparators.
0b0111	The implementation has seven Context Identifier Comparators.
0b1000	The implementation has eight Context Identifier Comparators.

All other values are reserved.

NUMSSCC, bits [23:20]

Indicates the number of Single-shot Comparator Controls that are available for tracing.

NUMSSCC	Meaning
0b0000	No Single-shot Comparator Controls are available.
0b0001	The implementation has one Single-shot Comparator Control.
0b0010	The implementation has two Single-shot Comparator Controls.
0b0011	The implementation has three Single-shot Comparator Controls.
0b0100	The implementation has four Single-shot Comparator Controls.
0b0101	The implementation has five Single-shot Comparator Controls.
0b0110	The implementation has six Single-shot Comparator Controls.
0b0111	The implementation has seven Single-shot Comparator Controls.
0b1000	The implementation has eight Single-shot Comparator Controls.

All other values are reserved.

NUMRSPAIR, bits [19:16]

Indicates the number of resource selector pairs that are available for tracing.

NUMRSPAIR	Meaning
0b0000	The implementation has zero resource selectors.
0b0001	The implementation has two resource selector pairs.
0b0010	The implementation has three resource selector pairs.
0b0011	The implementation has four resource selector pairs.
0b0100	The implementation has five resource selector pairs.
0b0101	The implementation has six resource selector pairs.
0b0110	The implementation has seven resource selector pairs.
0b0111	The implementation has eight resource selector pairs.
0b1000	The implementation has nine resource selector pairs.
0b1001	The implementation has ten resource selector pairs.
0b1010	The implementation has eleven resource selector pairs.
0b1011	The implementation has twelve resource selector pairs.
0b1100	The implementation has thirteen resource selector pairs.
0b1101	The implementation has fourteen resource selector pairs.
0b1110	The implementation has fifteen resource selector pairs.
0b1111	The implementation has sixteen resource selector pairs.

All other values are reserved.

NUMPC, bits [15:12]

Indicates the number of PE Comparator Inputs that are available for tracing.

NUMPC	Meaning
0b0000	No PE Comparator Inputs are available.
0b0001	The implementation has one PE Comparator Input.
0b0010	The implementation has two PE Comparator Inputs.
0b0011	The implementation has three PE Comparator Inputs.
0b0100	The implementation has four PE Comparator Inputs.
0b0101	The implementation has five PE Comparator Inputs.
0b0110	The implementation has six PE Comparator Inputs.
0b0111	The implementation has seven PE Comparator Inputs.
0b1000	The implementation has eight PE Comparator Inputs.

All other values are reserved.

Bits [11:9]

Reserved, RES0.

SUPPDAC, bit [8]

When TRCIDR4.NUMACPAIRS != 0b0000:

Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

SUPPDAC	Meaning
0b0	Data address comparisons not implemented.
0b1	Data address comparisons implemented.

This field reads as 0.

Otherwise:

Reserved, RES0.

NUMDVC, bits [7:4]

Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

NUMDVC	Meaning
0b0000	No data value comparators implemented.
0b0001	One data value comparator implemented.
0b0010	Two data value comparators implemented.
0b0011	Three data value comparators implemented.
0b0100	Four data value comparators implemented.
0b0101	Five data value comparators implemented.
0b0110	Six data value comparators implemented.
0b0111	Seven data value comparators implemented.
0b1000	Eight data value comparators implemented.

All other values are reserved.

This field reads as 0b0000.

NUMACPAIRS, bits [3:0]

Indicates the number of Address Comparator pairs that are available for tracing.

NUMACPAIRS	Meaning
0b0000	No Address Comparator pairs are available.
0b0001	The implementation has one Address Comparator pair.
0b0010	The implementation has two Address Comparator pairs.
0b0011	The implementation has three Address Comparator pairs.
0b0100	The implementation has four Address Comparator pairs.
0b0101	The implementation has five Address Comparator pairs.
0b0110	The implementation has six Address Comparator pairs.
0b0111	The implementation has seven Address Comparator pairs.
0b1000	The implementation has eight Address Comparator pairs.

All other values are reserved.

Accessing TRCIDR4

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

All other values are reserved.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0b000.

NUMSEQSTATE, bits [27:25]

Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented.

NUMSEQSTATE	Meaning
0b000	The Sequencer is not implemented.
0b100	Four Sequencer states are implemented.

All other values are reserved.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0b000.

Bit [24]

Reserved, RES0.

LPOVERRIDE, bit [23]

Indicates support for Low-power Override Mode.

LPOVERRIDE	Meaning
0b0	The trace unit does not support Low-power Override Mode.
0b1	The trace unit supports Low-power Override Mode.

ATBTRIG, bit [22]

Indicates if the implementation can support ATB triggers.

ATBTRIG	Meaning
0b0	The implementation does not support ATB triggers.
0b1	The implementation supports ATB triggers.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0.

TRACEIDSIZE, bits [21:16]

Indicates the trace ID width.

TRACEIDSIZE	Meaning
0b000000	The external trace interface is not implemented.
0b000111	The implementation supports a 7-bit trace ID.

All other values are reserved.

Note that AMBA ATB requires a 7-bit trace ID width.

Bits [15:12]

Reserved, RES0.

NUMEXTINSEL, bits [11:9]

Indicates how many External Input Selector resources are implemented.

NUMEXTINSEL	Meaning
0b000	No External Input Selector resources are available.
0b001	1 External Input Selector resource is available.
0b010	2 External Input Selector resources are available.
0b011	3 External Input Selector resources are available.
0b100	4 External Input Selector resources are available.

All other values are reserved.

NUMEXTIN, bits [8:0]

Indicates how many External Inputs are implemented.

NUMEXTIN	Meaning
0b11111111	Unified PMU event selection.

All other values are reserved.

Accessing TRCIDR5

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR6, ID Register 6

The TRCIDR6 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR6 bits [31:0] are architecturally mapped to External register [TRCIDR6\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR6 are UNDEFINED.

Attributes

TRCIDR6 is a 64-bit register.

Field descriptions

6362616059585756555453525150494847464544434241403938373635	34	33	32
RES0			
313029282726252423222120191817161514131211109876543	2	1	0
RES0			
EXLEVEL_RL_EL2		EXLEVEL_RL_EL1	EXLEVEL_RL_EL0

Bits [63:3]

Reserved, RES0.

EXLEVEL_RL_EL2, bit [2]

Indicates if Realm EL2 is implemented.

EXLEVEL_RL_EL2	Meaning
0b0	Realm EL2 is not implemented.
0b1	Realm EL2 is implemented.

EXLEVEL_RL_EL1, bit [1]

Indicates if Realm EL1 is implemented.

EXLEVEL_RL_EL1	Meaning
0b0	Realm EL1 is not implemented.
0b1	Realm EL1 is implemented.

EXLEVEL_RL_EL0, bit [0]

Indicates if Realm EL0 is implemented.

EXLEVEL_RL_EL0	Meaning
0b0	Realm EL0 is not implemented.
0b1	Realm EL0 is implemented.

Accessing TRCIDR6

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elseif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR6;
    elseif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elseif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = TRCIDR6;
    elseif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR6;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR7, ID Register 7

The TRCIDR7 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

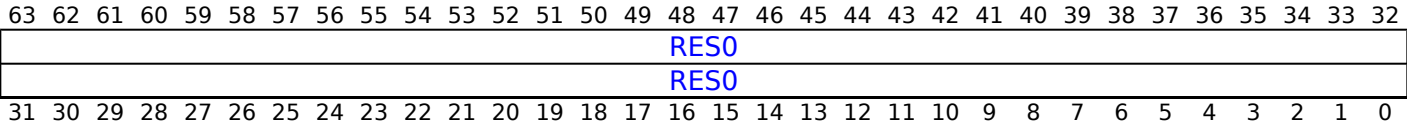
AArch64 System register TRCIDR7 bits [31:0] are architecturally mapped to External register [TRCIDR7\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR7 are UNDEFINED.

Attributes

TRCIDR7 is a 64-bit register.

Field descriptions



Bits [63:0]

Reserved, RES0.

Accessing TRCIDR7

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR7;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR7;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR7;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCIDR8, ID Register 8

The TRCIDR8 characteristics are:

Purpose

Returns the maximum speculation depth of the instruction trace element stream.

Configuration

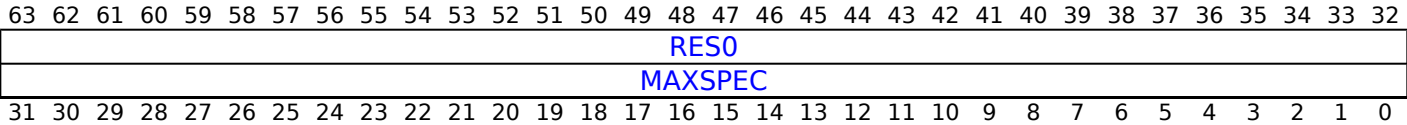
AArch64 System register TRCIDR8 bits [31:0] are architecturally mapped to External register [TRCIDR8\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCIDR8 are UNDEFINED.

Attributes

TRCIDR8 is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, RES0.

MAXSPEC, bits [31:0]

Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time.

Accessing TRCIDR8

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR8

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR9, ID Register 9

The TRCIDR9 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

AArch64 System register TRCIDR9 bits [31:0] are architecturally mapped to External register [TRCIDR9\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCIDR9 are UNDEFINED.

Attributes

TRCIDR9 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NUMPOKEY																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

NUMPOKEY, bits [31:0]

When TRCIDR0.TRCDATA != 0b00:

Indicates the number of P0 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR9

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIDR9

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR9;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIDR9;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR9;

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCIMSPEC0, IMP DEF Register 0

The TRCIMSPEC0 characteristics are:

Purpose

TRCIMSPEC0 shows the presence of any IMPLEMENTATION DEFINED features, and provides an interface to enable the features that are provided.

Configuration

AArch64 System register TRCIMSPEC0 bits [31:0] are architecturally mapped to External register [TRCIMSPEC0\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCIMSPEC0 are UNDEFINED.

Attributes

TRCIMSPEC0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																EN				SUPPORT											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:8]

Reserved, RES0.

EN, bits [7:4] When TRCIMSPEC0.SUPPORT != 0b0000:

Enable. Controls whether the IMPLEMENTATION DEFINED features are enabled.

EN	Meaning
0b0000	The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported.
0b0001	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0010	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0011	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0100	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0101	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0110	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0111	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1000	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1001	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1010	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1011	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1100	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1101	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1110	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1111	The trace unit behavior is IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Otherwise:

Reserved, RES0.

SUPPORT, bits [3:0]

Indicates whether the implementation supports IMPLEMENTATION DEFINED features.

SUPPORT	Meaning
0b0000	No IMPLEMENTATION DEFINED features are supported.
0b0001	IMPLEMENTATION DEFINED features are supported.
0b0010	IMPLEMENTATION DEFINED features are supported.
0b0011	IMPLEMENTATION DEFINED features are supported.
0b0100	IMPLEMENTATION DEFINED features are supported.
0b0101	IMPLEMENTATION DEFINED features are supported.
0b0110	IMPLEMENTATION DEFINED features are supported.
0b0111	IMPLEMENTATION DEFINED features are supported.
0b1000	IMPLEMENTATION DEFINED features are supported.
0b1001	IMPLEMENTATION DEFINED features are supported.
0b1010	IMPLEMENTATION DEFINED features are supported.
0b1011	IMPLEMENTATION DEFINED features are supported.
0b1100	IMPLEMENTATION DEFINED features are supported.
0b1101	IMPLEMENTATION DEFINED features are supported.
0b1110	IMPLEMENTATION DEFINED features are supported.
0b1111	IMPLEMENTATION DEFINED features are supported.

Use of nonzero values requires written permission from Arm.

Access to this field is **RO**.

Accessing TRCIMSPEC0

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIMSPEC0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCIMSPEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;

```

MSR TRCIMSPEC0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCIMSPEN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIMSPEC<n>, IMP DEF Register <n>, n = 1 - 7

The TRCIMSPEC<n> characteristics are:

Purpose

These registers might return information that is specific to an implementation, or enable features specific to an implementation to be programmed. The product Technical Reference Manual describes these registers.

Configuration

AArch64 System register TRCIMSPEC<n> bits [31:0] are architecturally mapped to External register [TRCIMSPEC<n>\[31:0\]](#).

This register is present only when the trace unit implements this OPTIONAL register, **FEAT_ETE is implemented** and **FEAT_TRC_SRRFEAT_ETE** is implemented. Otherwise, direct accesses to TRCIMSPEC<n> are UNDEFINED.

Attributes

TRCIMSPEC<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

This field reads as an IMPLEMENTATION DEFINED value and writes to this field have IMPLEMENTATION DEFINED behavior.

Accessing TRCIMSPEC<n>

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCIMSPEC<m> ; Where m = 1-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0:m[2:0]	0b111

```

integer m = UInt(CRm<2:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC[m];

```

MSR TRCIMSPEC<m>, <Xt> ; Where m = 1-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0:m[2:0]	0b111

```

integer m = UInt(CRm<2:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCIT, Trace Instrumentation

The TRCIT characteristics are:

Purpose

Generates an instrumentation packet in the trace.

Configuration

This instruction is present only when FEAT_ITE is implemented. Otherwise, direct accesses to TRCIT are UNDEFINED.

Attributes

TRCIT is a 64-bit System instruction.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																VALUE															
																VALUE															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:0]

Value to be included in the Instrumentation packet.

Executing TRCIT

Accesses to this instruction use the following encodings in the System instruction encoding space:

TRCIT <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0010	0b111

```

if PSTATE.EL == EL0 then
    AArch64.TRCIT(X[t, 64]);
elsif PSTATE.EL == EL1 then
    AArch64.TRCIT(X[t, 64]);
elsif PSTATE.EL == EL2 then
    AArch64.TRCIT(X[t, 64]);
elsif PSTATE.EL == EL3 then
    AArch64.TRCIT(X[t, 64]);

```

no old file

htmldiff from-

(new)

TRCITECR_EL1, Instrumentation Trace Control Register (EL1)

The TRCITECR_EL1 characteristics are:

Purpose

Provides EL1 controls for Trace Instrumentation.

Configuration

This register is present only when FEAT_ITE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCITECR_EL1 are UNDEFINED.

Attributes

TRCITECR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:2]

Reserved, RES0.

E1E, bit [1]

EL1 Instrumentation Trace Enable.

E1E	Meaning
0b0	Instrumentation trace prohibited at EL1.
0b1	Instrumentation trace not prohibited at EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

E0E, bit [0]

EL0 Instrumentation Trace Enable.

E0E	Meaning
0b0	Instrumentation trace prohibited at EL0.
0b1	Instrumentation trace not prohibited at EL0.

This field is ignored by the PE when EL2 is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRCITECR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCITECR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x888];
    else
        X[t, 64] = TRCITECR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TRCITECR_EL2;
    else
        X[t, 64] = TRCITECR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRCITECR_EL1;

```

MSR TRCITECR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x888] = X[t, 64];
    else
        TRCITECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TRCITECR_EL2 = X[t, 64];
    else
        TRCITECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRCITECR_EL1 = X[t, 64];

```

MRS <Xt>, TRCITECR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b011


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x888];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TRCITECR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = TRCITECR_EL1;
    else
        UNDEFINED;

```

MSR TRCITECR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x888] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TRCITECR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TRCITECR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRCITECR_EL2, Instrumentation Trace Control Register (EL2)

The TRCITECR_EL2 characteristics are:

Purpose

Provides EL2 controls for Trace Instrumentation.

Configuration

This register is present only when FEAT_ITE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCITECR_EL2 are UNDEFINED.

Attributes

TRCITECR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:2]

Reserved, RES0.

E2E, bit [1]

EL2 Instrumentation Trace Enable.

E2E	Meaning
0b0	Instrumentation trace prohibited at EL2.
0b1	Instrumentation trace not prohibited at EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

EOHE, bit [0]

EL0 Instrumentation Trace Enable.

EOHE	Meaning
0b0	Instrumentation trace prohibited at EL0 when HCR_EL2.TGE == 1.
0b1	Instrumentation trace not prohibited at EL0 when HCR_EL2.TGE == 1.

This field is ignored by the PE when any of the following are true:

- [HCR_EL2.TGE](#) == 0.
- EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRCITECR_EL2

When FEAT_VHE is implemented, and [HCR_EL2.E2H](#) is 1, without explicit synchronization, accesses from EL2 using the register name TRCITECR_EL2 or TRCITECR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCITECR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TRCITECR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRCITECR_EL2;

```

MSR TRCITECR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    TRCITECR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRCITECR_EL2 = X[t, 64];

```

MRS <Xt>, TRCITECR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x888];
    else
        X[t, 64] = TRCITECR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TRCITECR_EL2;
    else
        X[t, 64] = TRCITECR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRCITECR_EL1;

```

MSR TRCITECR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x888] = X[t, 64];
    else
        TRCITECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TRCITECR_EL2 = X[t, 64];
    else
        TRCITECR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRCITECR_EL1 = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TRCITEEDCR, Instrumentation Trace Extension External Debug Control Register

The TRCITEEDCR characteristics are:

Purpose

Controls instrumentation trace filtering.

Configuration

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and TRCIDR0.ITE == 1. Otherwise, direct accesses to TRCITEEDCR are UNDEFINED.

Attributes

TRCITEEDCR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32												
																RES0																											
																							RES0														RL	S	NS	E3	E2	E1	E0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												

Bits [63:7]

Reserved, RES0.

RL, bit [6] When FEAT_RME is implemented:

Instrumentation Trace in Realm state.

RL	Meaning
0b0	Instrumentation trace prohibited in Realm state.
0b1	Instrumentation trace permitted in Realm state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Realm state.

This field is ignored when SelfHostedTraceEnabled() returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S, bit [5]**When Secure state is implemented:**

Instrumentation Trace in Secure state.

S	Meaning
0b0	Instrumentation trace prohibited in Secure state.
0b1	Instrumentation trace permitted in Secure state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Secure state.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NS, bit [4]**When Any of Non-secure EL2, EL1, or EL0 are implemented:**

Instrumentation Trace in Non-secure state.

NS	Meaning
0b0	Instrumentation trace prohibited in Non-secure state.
0b1	Instrumentation trace permitted in Non-secure state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Non-secure state.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E3, bit [3]**When EL3 is implemented:**

Instrumentation Trace Enable at EL3.

E3	Meaning
0b0	Instrumentation trace prohibited at EL3.
0b1	Instrumentation trace permitted at EL3.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E<m>, bit [m], for m = 2 to 0

Instrumentation Trace Enable at EL<m>.

E<m>	Meaning
0b0	Instrumentation trace prohibited at EL<m>.
0b1	Instrumentation trace permitted at EL<m>.

Used in conjunction with [TRCCONFIGR.ITO](#), [TRCITEEDCR.NS](#), [TRCITEEDCR.S](#), and [TRCITEEDCR.RL](#) to control whether Instrumentation trace is permitted or prohibited at EL<m> in the specified security states.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

E<2> is RES0 if EL2 is not implemented in any security states.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCITEEDCR

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCITEEDCR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCITEEDCR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCITEEDCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCITEEDCR;

```

MSR TRCITEEDCR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDBGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCITEEDCR = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                TRCITEEDCR = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCITEEDCR = X[t, 64];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCOSLSR, Trace OS Lock Status Register

The TRCOSLSR characteristics are:

Purpose

Returns the status of the Trace OS Lock.

Configuration

AArch64 System register TRCOSLSR bits [31:0] are architecturally mapped to External register [TRCOSLSR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCOSLSR are UNDEFINED.

Attributes

TRCOSLSR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:5]

Reserved, RES0.

OSLM, bits [4:3, 0]

OS Lock model.

OSLM	Meaning
0b000	Trace OS Lock is not implemented.
0b010	Trace OS Lock is implemented.
0b100	Trace OS Lock is not implemented, and the trace unit is controlled by the PE OS Lock.

All other values are reserved.

This field reads as 0b100.

The OSLM field is split as follows:

- OSLM[2:1] is TRCOSLSR[4:3].
- OSLM[0] is TRCOSLSR[0].

Bit [2]

Reserved, RES0.

OSLK, bit [1]

OS Lock status.

OSLK	Meaning
0b0	The OS Lock is unlocked.
0b1	The OS Lock is locked.

Note that this field indicates the state of the PE OS Lock.

Accessing TRCOSLSR

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCOSLSR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0001	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCOSLSR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCOSLSR;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCOSLSR;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCOSLSR;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCPRGCTLR, Programming Control Register

The TRCPRGCTLR characteristics are:

Purpose

Enables the trace unit.

Configuration

AArch64 System register TRCPRGCTLR bits [31:0] are architecturally mapped to External register [TRCPRGCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCPRGCTLR are UNDEFINED.

Attributes

TRCPRGCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																RES0															EN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:1]

Reserved, RES0.

EN, bit [0]

Trace unit enable.

EN	Meaning
0b0	The trace unit is disabled.
0b1	The trace unit is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Accessing TRCPRGCTLR

Must be programmed.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCPRGCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCPRGCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCPRGCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCPRGCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCPRGCTLR;

```

MSR TRCPRGCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCPRGCTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCPRGCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCPRGCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCPRGCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCQCTLR, Q Element Control Register

The TRCQCTLR characteristics are:

Purpose

Controls when Q elements are enabled.

Configuration

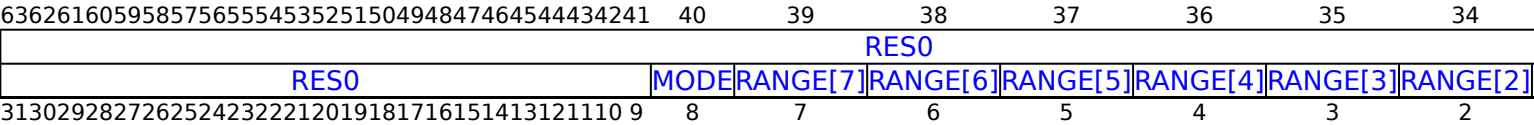
AArch64 System register TRCQCTLR bits [31:0] are architecturally mapped to External register [TRCQCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and TRCIDR0.QFILT == 1. Otherwise, direct accesses to TRCQCTLR are UNDEFINED.

Attributes

TRCQCTLR is a 64-bit register.

Field descriptions



Bits [63:9]

Reserved, RES0.

MODE, bit [8]

Selects whether the Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit is permitted to generate Q elements or address ranges where the trace unit is not permitted to generate Q elements:

MODE	Meaning
0b0	Exclude mode. The Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit must not generate Q elements. If no ranges are selected, Q elements are permitted across the entire memory map.
0b1	Include Mode. The Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit can generate Q elements. If all the implemented bits in RANGE are set to 0 then Q elements are disabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RANGE[<m>], bit [m], for m = 7 to 0

Specifies whether Address Range Comparator <m> controls Q elements.

RANGE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected.
0b1	The address range that Address Range Comparator <m> defines, is selected.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCQCTLR

Must be programmed if $\text{TRCCONFIGR.QE} \neq 0b00$.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCQCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCQCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCQCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCQCTLR;

```

MSR TRCQCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCQCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCQCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCQCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCRSCTLR<n>, Resource Selection Control Register <n>, n = 2 - 31

The TRCRSCTLR<n> characteristics are:

Purpose

Controls the selection of the resources in the trace unit.

Configuration

AArch64 System register TRCRSCTLR<n> bits [31:0] are architecturally mapped to External register [TRCRSCTLR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $(\text{UInt}(\text{TRCIDR4.NUMRSPAIR}) + 1) * 2 > n$. Otherwise, direct accesses to TRCRSCTLR<n> are UNDEFINED.

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

TRCRSCTLR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0										PAIRINV		INV		GROUP				SELECT													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:22]

Reserved, RES0.

PAIRINV, bit [21]

When n MOD 2 == 0:

Controls whether the combined result from a resource selector pair is inverted.

PAIRINV	Meaning
0b0	Do not invert the combined output of the 2 resource selectors.
0b1	Invert the combined output of the 2 resource selectors.

If:

- A is the register TRCRSCTLR<n>.
- B is the register TRCRSCTLR<n+1>.

Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows:

- 0b000 -> A and B.
- 0b001 -> Reserved.
- 0b010 -> not(A) and B.
- 0b011 -> not(A) and not(B).
- 0b100 -> not(A) or not(B).
- 0b101 -> not(A) or B.
- 0b110 -> Reserved.
- 0b111 -> A or B.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

INV, bit [20]

Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted.

INV	Meaning
0b0	Do not invert the output of this selector.
0b1	Invert the output of this selector.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

GROUP, bits [19:16]

Selects a group of resources.

GROUP	Meaning	SELECT
0b0000	External Input Selectors.	SELECT encoding for External Input Selectors
0b0001	PE Comparator Inputs.	SELECT encoding for PE Comparator Inputs
0b0010	Counters and Sequencer.	SELECT encoding for Counters and Sequencer
0b0011	Single-shot Comparator Controls.	SELECT encoding for Single-shot Comparator Controls
0b0100	Single Address Comparators.	SELECT encoding for Single Address Comparators
0b0101	Address Range Comparators.	SELECT encoding for Address Range Comparators
0b0110	Context Identifier Comparators.	SELECT encoding for Context Identifier Comparators
0b0111	Virtual Context Identifier Comparators.	SELECT encoding for Virtual Context Identifier Comparators

All other values are reserved.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT, bits [15:0]

Resource Specific Controls. Contains the controls specific to the resource group selected by GROUP, described in the following sections.

SELECT encoding for External Input Selectors

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												EXTIN[3]	EXTIN[2]	EXTIN[1]	EXTIN[0]

Bits [15:4]

Reserved, RES0.

EXTIN[<m>], bit [m], for m = 3 to 0

Selects one or more External Inputs.

EXTIN[<m>]	Meaning
0b0	Ignore EXTIN <m>.
0b1	Select EXTIN <m>.

This bit is RES0 if m >= [TRCIDR5](#).NUMEXTINSEL.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for PE Comparator Inputs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								PECOMP[7]	PECOMP[6]	PECOMP[5]	PECOMP[4]	PECOMP[3]	PECOMP[2]	PECOMP[1]	PECOMP[0]

Bits [15:8]

Reserved, RES0.

PECOMP[<m>], bit [m], for m = 7 to 0

Selects one or more PE Comparator Inputs.

PECOMP[<m>]	Meaning
0b0	Ignore PE Comparator Input <m>.
0b1	Select PE Comparator Input <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMPC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Counters and Sequencer

15	14	13	12	11	10	9	8	7	6	5	4	3	2
RES0								SEQUENCER[3]	SEQUENCER[2]	SEQUENCER[1]	SEQUENCER[0]	COUNTERS[3]	COUNTERS[2]

Bits [15:8]

Reserved, RES0.

SEQUENCER[<m>], bit [m+4], for m = 3 to 0

Sequencer states.

SEQUENCER[<m>]	Meaning
0b0	Ignore Sequencer state <m>.
0b1	Select Sequencer state <m>.

This bit is RES0 if m >= [TRCIDR5](#).NUMSEQSTATE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

COUNTERS[<m>], bit [m], for m = 3 to 0

Counters resources at zero.

COUNTERS[<m>]	Meaning
0b0	Ignore Counter <m>.
0b1	Select Counter <m> is zero.

This bit is RES0 if m >= [TRCIDR5](#).NUMCNTR.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Single-shot Comparator Controls

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	SINGLE_SHOT[7]	SINGLE_SHOT[6]	SINGLE_SHOT[5]	SINGLE_SHOT[4]	SINGLE_SHOT[3]	SINGLE_SHOT[2]	SINGLE_SHOT[1]	SINGLE_SHOT[0]	SINGLE_SHOT[7]	SINGLE_SHOT[6]	SINGLE_SHOT[5]	SINGLE_SHOT[4]	SINGLE_SHOT[3]	SINGLE_SHOT[2]	SINGLE_SHOT[1]

Bits [15:8]

Reserved, RES0.

SINGLE_SHOT[<m>], bit [m], for m = 7 to 0

Selects one or more Single-shot Comparator Controls.

SINGLE_SHOT[<m>]	Meaning
0b0	Ignore Single-shot Comparator Control <m>.
0b1	Select Single-shot Comparator Control <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMSSCC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Single Address Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAC[15]	SAC[14]	SAC[13]	SAC[12]	SAC[11]	SAC[10]	SAC[9]	SAC[8]	SAC[7]	SAC[6]	SAC[5]	SAC[4]	SAC[3]	SAC[2]	SAC[1]	SAC[0]

SAC[<m>], bit [m], for m = 15 to 0

Selects one or more Single Address Comparators.

SAC[<m>]	Meaning
0b0	Ignore Single Address Comparator <m>.
0b1	Select Single Address Comparator <m>.

This bit is RES0 if m >= 2 × [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Address Range Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								ARC[7]	ARC[6]	ARC[5]	ARC[4]	ARC[3]	ARC[2]	ARC[1]	ARC[0]

Bits [15:8]

Reserved, RES0.

ARC[<m>], bit [m], for m = 7 to 0

Selects one or more Address Range Comparators.

ARC[<m>]	Meaning
0b0	Ignore Address Range Comparator <m>.
0b1	Select Address Range Comparator <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Context Identifier Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								CID[7]	CID[6]	CID[5]	CID[4]	CID[3]	CID[2]	CID[1]	CID[0]

Bits [15:8]

Reserved, RES0.

CID[<m>], bit [m], for m = 7 to 0

Selects one or more Context Identifier Comparators.

CID[<m>]	Meaning
0b0	Ignore Context Identifier Comparator <m>.
0b1	Select Context Identifier Comparator <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMCIDC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Virtual Context Identifier Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								VMID[7]	VMID[6]	VMID[5]	VMID[4]	VMID[3]	VMID[2]	VMID[1]	VMID[0]

Bits [15:8]

Reserved, RES0.

VMID[<m>], bit [m], for m = 7 to 0

Selects one or more Virtual Context Identifier Comparators.

VMID[<m>]	Meaning
0b0	Ignore Virtual Context Identifier Comparator <m>.
0b1	Select Virtual Context Identifier Comparator <m>.

This bit is RES0 if m >= [TRCIDR4.NUMVMIDC](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCRSCTLR<n>

Must be programmed if any of the following are true:

- [TRCCNTCTLR<a>.RLDEVENT](#).TYPE == 0 and [TRCCNTCTLR<a>.RLDEVENT](#).SEL == n.
- [TRCCNTCTLR<a>.RLDEVENT](#).TYPE == 1 and [TRCCNTCTLR<a>.RLDEVENT](#).SEL == n/2.
- [TRCCNTCTLR<a>.CNTEVENT](#).TYPE == 0 and [TRCCNTCTLR<a>.CNTEVENT](#).SEL == n.
- [TRCCNTCTLR<a>.CNTEVENT](#).TYPE == 1 and [TRCCNTCTLR<a>.CNTEVENT](#).SEL == n/2.
- [TRCEVENTCTL0R.EVENT0](#).TYPE == 0 and [TRCEVENTCTL0R.EVENT0](#).SEL == n.
- [TRCEVENTCTL0R.EVENT0](#).TYPE == 1 and [TRCEVENTCTL0R.EVENT0](#).SEL == n/2.
- [TRCEVENTCTL0R.EVENT1](#).TYPE == 0 and [TRCEVENTCTL0R.EVENT1](#).SEL == n.
- [TRCEVENTCTL0R.EVENT1](#).TYPE == 1 and [TRCEVENTCTL0R.EVENT1](#).SEL == n/2.
- [TRCEVENTCTL0R.EVENT2](#).TYPE == 0 and [TRCEVENTCTL0R.EVENT2](#).SEL == n.
- [TRCEVENTCTL0R.EVENT2](#).TYPE == 1 and [TRCEVENTCTL0R.EVENT2](#).SEL == n/2.
- [TRCEVENTCTL0R.EVENT3](#).TYPE == 0 and [TRCEVENTCTL0R.EVENT3](#).SEL == n.
- [TRCEVENTCTL0R.EVENT3](#).TYPE == 1 and [TRCEVENTCTL0R.EVENT3](#).SEL == n/2.
- [TRCSEQEVR<a>.B](#).TYPE == 0 and [TRCSEQEVR<a>.B](#).SEL == n.
- [TRCSEQEVR<a>.B](#).TYPE == 1 and [TRCSEQEVR<a>.B](#).SEL == n/2.
- [TRCSEQEVR<a>.F](#).TYPE == 0 and [TRCSEQEVR<a>.F](#).SEL == n.
- [TRCSEQEVR<a>.F](#).TYPE == 1 and [TRCSEQEVR<a>.F](#).SEL == n/2.
- [TRCSEQRSTEVR.RST](#).TYPE == 0 and [TRCSEQRSTEVR.RST](#).SEL == n.
- [TRCSEQRSTEVR.RST](#).TYPE == 1 and [TRCSEQRSTEVR.RST](#).SEL == n/2.
- [TRCTSCTLR.EVENT](#).TYPE == 0 and [TRCTSCTLR.EVENT](#).SEL == n.
- [TRCTSCTLR.EVENT](#).TYPE == 1 and [TRCTSCTLR.EVENT](#).SEL == n/2.
- [TRCVICTLR.EVENT](#).TYPE == 0 and [TRCVICTLR.EVENT](#).SEL == n.
- [TRCVICTLR.EVENT](#).TYPE == 1 and [TRCVICTLR.EVENT](#).SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCRSCTLR<m> ; Where m = 2-31

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	m[3:0]	0b00:m[4]


```

integer m = UInt(op2<0>:CRm<3:0>);

if m >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCRSCTLR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCRSCTLR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[m];

```

MSR TRCRSCTLR<m>, <Xt> ; Where m = 2-31

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	m[3:0]	0b00:m[4]

```

integer m = UInt(op2<0>:CRm<3:0>);

if m >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCRSR, Resources Status Register

The TRCRSR characteristics are:

Purpose

Use this to set, or read, the status of the resources.

Configuration

AArch64 System register TRCRSR bits [31:0] are architecturally mapped to External register [TRCRSR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCRSR are UNDEFINED.

Attributes

TRCRSR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33							
																					RES0																
RES0																					TA	EVENT[3]		EVENT[2]		EVENT[1]		EVENT[0]		RES0		EXTIN[3]		EXTIN[2]		EXTIN[1]	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1							

Bits [63:13]

Reserved, RES0.

TA, bit [12]

Tracing active.

TA	Meaning
0b0	Tracing is not active.
0b1	Tracing is active.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

EVENT[<m>], bit [m+8], for m = 3 to 0

Untraced status of ETEEvents.

EVENT[<m>]	Meaning
0b0	An ETEEvent <m> has not occurred.
0b1	An ETEEvent <m> has occurred while the resources were in the Paused state.

This bit is RES0 if [TRCIDR4](#).NUMRSPAIR == 0 || m > [TRCIDR0](#).NUMEVENT.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [7:4]

Reserved, RES0.

EXTIN[<m>], bit [m], for m = 3 to 0

The sticky status of the External Input Selectors.

EXTIN[<m>]	Meaning
0b0	An event selected by External Input Selector <m> has not occurred.
0b1	At least one event selected by External Input Selector <m> has occurred while the resources were in the Paused state.

This bit is RES0 if m >= [TRCIDR5.NUMEXTINSEL](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCRSR

Must always be programmed.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCRSR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCRSR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCRSR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSR;

```

MSR TRCRSR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCRSR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCRSR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSEQEVR<n>, Sequencer State Transition Control Register <n>, n = 0 - 2

The TRCSEQEVR<n> characteristics are:

Purpose

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configuration

AArch64 System register TRCSEQEVR<n> bits [31:0] are architecturally mapped to External register [TRCSEQEVR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQEVR<n> are UNDEFINED.

Attributes

TRCSEQEVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																B_TYPE	RES0	B_SEL				F_TYPE	RES0	F_SEL							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

B_TYPE, bit [15]

Chooses the type of Resource Selector.

Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == 0x14 then when event 0x14 occurs, the Sequencer moves from state 3 to state 2.

B_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [14:13]

Reserved, RES0.

B_SEL, bits [12:8]

Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Backward field. Selects the single Resource Selector or Resource Selector pair.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

F_TYPE, bit [7]

Chooses the type of Resource Selector.

Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2.

F_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

F_SEL, bits [4:0]

Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Forward field. Selects the single Resource Selector or Resource Selector pair.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSEQEVR<n>

Must be programmed if [TRCRSCTLR<a>](#).GROUP == 0b0010 and [TRCRSCTLR<a>](#).SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSEQEVR<m> ; Where m = 0-2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b00:m[1:0]	0b100

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSEQEVR[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSEQEVR[m];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSEQEVR[m];
```

MSR TRCSEQEVR<m>, <Xt> ; Where m = 0-2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b00:m[1:0]	0b100

```

integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQEVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                TRCSEQEVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQEVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSEQRSTEV, Sequencer Reset Control Register

The TRCSEQRSTEV characteristics are:

Purpose

Moves the Sequencer to state 0 when a programmed resource event occurs.

Configuration

AArch64 System register TRCSEQRSTEV bits [31:0] are architecturally mapped to External register [TRCSEQRSTEV\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQRSTEV are UNDEFINED.

Attributes

TRCSEQRSTEV is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40		39		38	37	36	35	34	33	32																
RES0																																																	
RES0																								RST_TYPE		RES0		RST_SEL																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8		7		6	5	4	3	2	1	0																

Bits [63:8]

Reserved, RES0.

RST_TYPE, bit [7]

Chooses the type of Resource Selector.

RST_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQRSTEV.RST.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQRSTEV.RST.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQRSTEV.RST.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

RST_SEL, bits [4:0]

Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQRSTEV.RST.TYPE controls whether TRCSEQRSTEV.RST.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSEQRSTEV

Must be programmed if [TRCRSCTLR<a>.GROUP == 0b0010](#) and [TRCRSCTLR<a>.SEQUENCER != 0b0000](#).

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSEQRSTEV

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSEQRSTEV;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSEQRSTEV;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQRSTEV;

```

MSR TRCSEQRSTEV, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQRSTEV = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQRSTEV = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQRSTEV = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSEQSTR, Sequencer State Register

The TRCSEQSTR characteristics are:

Purpose

Use this to set, or read, the Sequencer state.

Configuration

AArch64 System register TRCSEQSTR bits [31:0] are architecturally mapped to External register [TRCSEQSTR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQSTR are UNDEFINED.

Attributes

TRCSEQSTR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															STATE
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:2]

Reserved, RES0.

STATE, bits [1:0]

Set or returns the state of the Sequencer.

STATE	Meaning
0b00	State 0.
0b01	State 1.
0b10	State 2.
0b11	State 3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSEQSTR

Must be programmed if [TRCRSCTLR<a>](#).GROUP == 0b0010 and [TRCRSCTLR<a>](#).SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSEQSTR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0111	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCSQSTR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSQSTR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSQSTR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSQSTR;

```

MSR TRCSQSTR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0111	0b100


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCSEQSTR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQSTR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSEQSTR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQSTR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSSCCR<n>, Single-shot Comparator Control Register <n>, n = 0 - 7

The TRCSSCCR<n> characteristics are:

Purpose

Controls the corresponding Single-shot Comparator Control resource.

Configuration

AArch64 System register TRCSSCCR<n> bits [31:0] are architecturally mapped to External register [TRCSSCCR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and UInt(TRCIDR4.NUMSSCC) > n. Otherwise, direct accesses to TRCSSCCR<n> are UNDEFINED.

Attributes

TRCSSCCR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43			
																			RES0				
RES0							RST	ARC[7]	ARC[6]	ARC[5]	ARC[4]	ARC[3]	ARC[2]	ARC[1]	ARC[0]	SAC[15]	SAC[14]	SAC[13]	SAC[12]	SAC[11]			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11			

Bits [63:25]

Reserved, RES0.

RST, bit [24]

Selects the Single-shot Comparator Control mode.

RST	Meaning
0b0	The Single-shot Comparator Control is in single-shot mode.
0b1	The Single-shot Comparator Control is in multi-shot mode.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

ARC[<m>], bit [m+16], for m = 7 to 0

Selects one or more Address Range Comparators for Single-shot control.

ARC[<m>]	Meaning
0b0	The Address Range Comparator <m>, is not selected for Single-shot control.
0b1	The Address Range Comparator <m>, is selected for Single-shot control.

This bit is RES0 if m >= [TRCIDR4.NUMACPAIRS](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SAC[<m>], bit [m], for m = 15 to 0

Selects one or more Single Address Comparators for Single-shot control.

SAC[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected for Single-shot control.
0b1	The Single Address Comparator <m>, is selected for Single-shot control.

This bit is RES0 if m >= 2 × [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSSCCR<n>

Must be programmed if any [TRCRSCTLR<a>](#).GROUP == 0b0011 and [TRCRSCTLR<a>](#).SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSSCCR<m> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0:m[2:0]	0b010

```

integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSSCCR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSSCCR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCCR[m];

```

MSR TRCSSCCR<m>, <Xt> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0:m[2:0]	0b010

```

integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCCR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCCR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCCR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSSCSR<n>, Single-shot Comparator Control Status Register <n>, n = 0 - 7

The TRCSSCSR<n> characteristics are:

Purpose

Returns the status of the corresponding Single-shot Comparator Control.

Configuration

AArch64 System register TRCSSCSR<n> bits [31:0] are architecturally mapped to External register [TRCSSCSR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $UInt(TRCIDR4.NUMSSCC) > n$. Otherwise, direct accesses to TRCSSCSR<n> are UNDEFINED.

Attributes

TRCSSCSR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
STATUS	PENDING	RES0																											PCDV	DA	INST
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

STATUS, bit [31]

Single-shot Comparator Control status. Indicates if any of the comparators selected by this Single-shot Comparator control have matched. The selected comparators are defined by [TRCSSCCR<n> .ARC](#), [TRCSSCCR<n> .SAC](#), and [TRCSSPCICR<n> .PC](#).

STATUS	Meaning
0b0	No match has occurred. When the first match occurs, this field takes a value of 1. It remains at 1 until explicitly modified by a write to this register.
0b1	One or more matches has occurred. If TRCSSCCR<n> .RST == 0 then: <ul style="list-style-type: none"> There is only one match and no more matches are possible. Software must reset this field to 0 to re-enable the Single-shot Comparator Control.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

PENDING, bit [30]

Single-shot pending status. The Single-shot Comparator Control fired while the resources were in the Paused state.

PENDING	Meaning
0b0	No match has occurred.
0b1	One or more matches has occurred.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [29:4]

Reserved, RES0.

PC, bit [3]

PE Comparator Input support. Indicates if the Single-shot Comparator Control supports PE Comparator Inputs.

PC	Meaning
0b0	This Single-shot Comparator Control does not support PE Comparator Inputs. Selecting any PE Comparator Inputs using the associated TRCSSPCICR<n> results in CONSTRAINED UNPREDICTABLE behavior of the Single-shot Comparator Control resource. The Single-shot Comparator Control might match unexpectedly or might not match.
0b1	This Single-shot Comparator Control supports PE Comparator Inputs.

Access to this field is **RO**.

DV, bit [2]

Data value comparator support. Data value comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

DV	Meaning
0b0	This Single-shot Comparator Control does not support data value comparisons.
0b1	This Single-shot Comparator Control supports data value comparisons.

This field reads as 0.

Access to this field is **RO**.

DA, bit [1]

Data Address Comparator support. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

DA	Meaning
0b0	This Single-shot Comparator Control does not support data address comparisons.
0b1	This Single-shot Comparator Control supports data address comparisons.

This field reads as 0.

Access to this field is **RO**.

INST, bit [0]

Instruction Address Comparator support. Indicates if the Single-shot Comparator Control supports instruction address comparisons.

INST	Meaning
0b0	This Single-shot Comparator Control does not support instruction address comparisons.
0b1	This Single-shot Comparator Control supports instruction address comparisons.

This field reads as 1.

Access to this field is **RO**.

Accessing TRCSSCSR<n>

Must be programmed if [TRCRSCTLR<a>.GROUP](#) == 0b0011 and [TRCRSCTLR<a>.SINGLE_SHOT\[n\]](#) == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSSCSR<m> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1:m[2:0]	0b010


```

integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCSSCSRn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSSCSR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSSCSR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCSR[m];

```

MSR TRCSSCSR<m>, <Xt> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1:m[2:0]	0b010

```

integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCSSCSRn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCSR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCSR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSCSR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCSSPCICR<n> characteristics are:

Purpose

Returns the status of the corresponding Single-shot Comparator Control.

Configuration

AArch64 System register TRCSSPCICR<n> bits [31:0] are architecturally mapped to External register [TRCSSPCICR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, $\text{UInt}(\text{TRCIDR4.NUMSSCC}) > n$, $\text{UInt}(\text{TRCIDR4.NUMPC}) > 0$ and $\text{TRCSSCSR}\langle n \rangle.\text{PC} == 1$. Otherwise, direct accesses to $\text{TRCSSPCICR}\langle n \rangle$ are UNDEFINED.

Attributes

TRCSSPCICR<n> is a 64-bit register.

Field descriptions

Diagram illustrating the structure of a 64-bit register. The register is divided into two 32-bit halves. The upper half is labeled **RES0** and the lower half is labeled **RES0**. The lower half is further divided into eight 4-bit fields labeled **PC[7]**, **PC[6]**, **PC[5]**, **PC[4]**, **PC[3]**, **PC[2]**, **PC[1]**, and **PC[0]**. Bit positions 63 down to 32 are shown above the register, and bit positions 31 down to 0 are shown below the register.

Bits [63:8]

Reserved, RES0.

PC[<m>], bit [m], for m = 7 to 0

Selects one or more PE Comparator Inputs for Single-shot control.

PC[<m>]	Meaning
0b0	The single PE Comparator Input <m>, is not selected as for Single-shot control.
0b1	The single PE Comparator Input <m>, is selected as for Single-shot control.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMPC}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSSPCICR<n>

Must be programmed if implemented and any [TRCRSCTLR<a>](#).GROUP == 0b0011 and [TRCRSCTLR<a>](#).SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSSPCICR<m> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0:m[2:0]	0b011

```
integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSPCICR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSPCICR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSPCICR[m];
```

MSR TRCSSPCICR<m>, <Xt> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0:m[2:0]	0b011

```

integer m = UInt(CRm<2:0>);

if m >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSPCICR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSPCICR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSSPCICR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSTALLCTLR, Stall Control Register

The TRCSTALLCTLR characteristics are:

Purpose

Enables trace unit functionality that prevents trace unit buffer overflows.

Configuration

AArch64 System register TRCSTALLCTLR bits [31:0] are architecturally mapped to External register [TRCSTALLCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and TRCIDR3.STALLCTL == 1. Otherwise, direct accesses to TRCSTALLCTLR are UNDEFINED.

Attributes

TRCSTALLCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																		RES0																			
RES0																		NOOVERFLOW				RES0				ISTALL		RES0				LEVEL					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

Bits [63:14]

Reserved, RES0.

NOOVERFLOW, bit [13]

When TRCIDR3.NOOVERFLOW == 1:

Trace overflow prevention.

NOOVERFLOW	Meaning
0b0	Trace unit buffer overflow prevention is disabled.
0b1	Trace unit buffer overflow prevention is enabled.

Note that enabling this feature might cause a significant performance impact.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

ISTALL, bit [8]

Instruction stall control. Controls if a trace unit can stall the PE when the trace buffer space is less than LEVEL.

ISTALL	Meaning
0b0	The trace unit must not stall the PE.
0b1	The trace unit can stall the PE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [7:4]

Reserved, RES0.

LEVEL, bits [3:0]

Threshold level field. The field can support 16 monotonic levels from 0b0000 to 0b1111.

The value 0b0000 defines the Minimal invasion level. This setting has a greater risk of a trace unit buffer overflow.

The value 0b1111 defines the Maximum invasion level. This setting has a reduced risk of a trace unit buffer overflow.

Note that for some implementations, invasion might occur at the minimal invasion level.

One or more of the least significant bits of LEVEL are permitted to be RES0. Arm recommends that LEVEL[3:2] are fully implemented. Arm strongly recommends that LEVEL[3] is always implemented. If one or more bits are RES0 and are written with a non-zero value, the effective value of LEVEL is rounded down to the nearest power of 2 value which has the RES0 bits as zero. For example, if LEVEL[1:0] are RES0 and a value of 0b1110 is written to LEVEL, the effective value of LEVEL is 0b1100.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSTALLCTLR

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSTALLCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSTALLCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSTALLCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSTALLCTLR;

```

MSR TRCSTALLCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSTALLCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSTALLCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSTALLCTLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSTATR, Trace Status Register

The TRCSTATR characteristics are:

Purpose

Returns the trace unit status.

Configuration

AArch64 System register TRCSTATR bits [31:0] are architecturally mapped to External register [TRCSTATR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_SR is **implemented**. Otherwise, direct accesses to TRCSTATR are UNDEFINED.

Attributes

TRCSTATR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34		33		32
RES0																																	
RES0																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2		1		0

Bits [63:2]

Reserved, RES0.

PMSTABLE, bit [1]

Programmers' model stable.

PMSTABLE	Meaning
0b0	The programmers' model is not stable.
0b1	The programmers' model is stable.

Accessing this field has the following behavior:

- When the trace unit is enabled, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

IDLE, bit [0]

Idle status.

IDLE	Meaning
0b0	The trace unit is not idle.
0b1	The trace unit is idle.

Accessing TRCSTATR

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSTATR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCSTATR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSTATR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSTATR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSTATR;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSYNCP, Synchronization Period Register

The TRCSYNCP characteristics are:

Purpose

Controls how often trace protocol synchronization requests occur.

Configuration

AArch64 System register TRCSYNCP bits [31:0] are architecturally mapped to External register [TRCSYNCP\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCSYNCP are UNDEFINED.

Attributes

TRCSYNCP is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:5]

Reserved, RES0.

PERIOD, bits [4:0]

Defines the number of bytes of trace between each periodic trace protocol synchronization request.

PERIOD	Meaning
0b00000	Trace protocol synchronization is disabled.
0b01000	Trace protocol synchronization request occurs after 2^8 bytes of trace.
0b01001	Trace protocol synchronization request occurs after 2^9 bytes of trace.
0b01010	Trace protocol synchronization request occurs after 2^{10} bytes of trace.
0b01011	Trace protocol synchronization request occurs after 2^{11} bytes of trace.
0b01100	Trace protocol synchronization request occurs after 2^{12} bytes of trace.
0b01101	Trace protocol synchronization request occurs after 2^{13} bytes of trace.
0b01110	Trace protocol synchronization request occurs after 2^{14} bytes of trace.
0b01111	Trace protocol synchronization request occurs after 2^{15} bytes of trace.
0b10000	Trace protocol synchronization request occurs after 2^{16} bytes of trace.
0b10001	Trace protocol synchronization request occurs after 2^{17} bytes of trace.
0b10010	Trace protocol synchronization request occurs after 2^{18} bytes of trace.
0b10011	Trace protocol synchronization request occurs after 2^{19} bytes of trace.
0b10100	Trace protocol synchronization request occurs after 2^{20} bytes of trace.

Other values are reserved. If a reserved value is programmed into PERIOD, then the behavior of the synchronization period counter is CONSTRAINED UNPREDICTABLE and one of the following behaviors occurs:

- No trace protocol synchronization requests are generated by this counter.
- Trace protocol synchronization requests occur at the specified period.
- Trace protocol synchronization requests occur at some other UNKNOWN period which can vary.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSYNCP

Must be programmed if [TRCIDR3.SYNCP](#) == 0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCSYNCP

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSYNCP;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCSYNCP;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSYNCP;

```

MSR TRCSYNCP, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSYNCP = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCSYNCP = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSYNCP = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCTRACEIDR, Trace ID Register

The TRCTRACEIDR characteristics are:

Purpose

Sets the trace ID for instruction trace.

Configuration

AArch64 System register TRCTRACEIDR bits [31:0] are architecturally mapped to External register [TRCTRACEIDR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented and FEAT_TRC_SR is implemented**. Otherwise, direct accesses to TRCTRACEIDR are UNDEFINED.

Attributes

TRCTRACEIDR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																RES0																					
RES0																										TRACEID											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

Bits [63:7]

Reserved, RES0.

TRACEID, bits [6:0]

Trace ID field. Sets the trace ID value for instruction trace. The width of the field is indicated by the value of [TRCIDR5.TRACEIDSIZE](#). Unimplemented bits are RES0.

If an implementation supports AMBA ATB, then:

- The width of the field is 7 bits.
- Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure.

See the AMBA ATB Protocol Specification for information about which ATID values are reserved.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCTRACEIDR

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCTRACEIDR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCTRACEIDR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCTRACEIDR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCTRACEIDR;

```

MSR TRCTRACEIDR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCTRAIDR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCTRAIDR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCTRAIDR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCTSCTLR, Timestamp Control Register

The TRCTSCTLR characteristics are:

Purpose

Controls the insertion of global timestamps in the trace stream.

Configuration

AArch64 System register TRCTSCTLR bits [31:0] are architecturally mapped to External register [TRCTSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and TRCIDR0.TSSIZE != 0b00000. Otherwise, direct accesses to TRCTSCTLR are UNDEFINED.

Attributes

TRCTSCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40		39		38	37	36	35	34	33	32
RES0																																	
RES0																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8		7		6	5	4	3	2	1	0
																								EVENT_TYPE				RES0		EVENT_SEL			

Bits [63:8]

Reserved, RES0.

EVENT_TYPE, bit [7] When TRCIDR4.NUMRSPAIR != 0b0000:

Chooses the type of Resource Selector.

EVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCTSCTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCTSCTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCTSCTLR.EVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [6:5]

Reserved, RES0.

EVENT_SEL, bits [4:0]
When TRCIDR4.NUMRSPAIR != 0b0000:

Defines the selected Resource Selector or pair of Resource Selectors. TRCTSCTLR.EVENT.TYPE controls whether TRCTSCTLR.EVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCTSCTLR

Must be programmed if [TRCCONFIGR](#).TS == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCTSCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCTSCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCTSCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCTSCTLR;

```

MSR TRCTSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDBGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCTSCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCTSCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCTSCTLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCVICTLR characteristics are:

Purpose

Controls instruction trace filtering.

Configuration

AArch64 System register TRCVICTLR bits [31:0] are architecturally mapped to External register [TRCVICTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_SR is implemented. Otherwise, direct accesses to TRCVICTLR are UNDEFINED.

Attributes

TRCVICTLR is a 64-bit register.

Field descriptions

6362616059										58										57										56										55										54										53										52									
RES0										EXLEVEL_RL_EL2										EXLEVEL_RL_EL1										EXLEVEL_RL_EL0										RES0										EXLEVEL_NS_EL2										EXLEVEL_NS_EL1										EXLEVEL_NS_EL0									
3130292827										26										25										24										23										22										21										20									

Bits [63:27]

Reserved, RES0.

EXLEVEL_RL_EL2, bit [26]

When TRCIDR6.EXLEVEL RL_EL2 == 1:

Filter instruction trace for EL2 in Realm state.

EXLEVEL_RL_EL2	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL2 is 0 the trace unit generates instruction trace for EL2 in Realm state.
	When TRCVICTLR.EXLEVEL_NS_EL2 is 1 the trace unit does not generate instruction trace for EL2 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL2 is 0 the trace unit does not generate instruction trace for EL2 in Realm state.
	When TRCVICTLR.EXLEVEL_NS_EL2 is 1 the trace unit generates instruction trace for EL2 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL1, bit [25]**When TRCIDR6.EXLEVEL_RL_EL1 == 1:**

Filter instruction trace for EL1 in Realm state.

EXLEVEL_RL_EL1	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL1 is 0 the trace unit generates instruction trace for EL1 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL1 is 1 the trace unit does not generate instruction trace for EL1 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL1 is 0 the trace unit does not generate instruction trace for EL1 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL1 is 1 the trace unit generates instruction trace for EL1 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL0, bit [24]**When TRCIDR6.EXLEVEL_RL_EL0 == 1:**

Filter instruction trace for EL0 in Realm state.

EXLEVEL_RL_EL0	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL0 is 0 the trace unit generates instruction trace for EL0 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL0 is 1 the trace unit does not generate instruction trace for EL0 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL0 is 0 the trace unit does not generate instruction trace for EL0 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL0 is 1 the trace unit generates instruction trace for EL0 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [23]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [22]**When Non-secure EL2 is implemented:**

Filter instruction trace for EL2 in Non-secure state.

EXLEVEL_NS_EL2	Meaning
0b0	The trace unit generates instruction trace for EL2 in Non-secure state.
0b1	The trace unit does not generate instruction trace for EL2 in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL1, bit [21]**When Non-secure EL1 is implemented:**

Filter instruction trace for EL1 in Non-secure state.

EXLEVEL_NS_EL1	Meaning
0b0	The trace unit generates instruction trace for EL1 in Non-secure state.
0b1	The trace unit does not generate instruction trace for EL1 in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_ELO, bit [20]**When Non-secure ELO is implemented:**

Filter instruction trace for ELO in Non-secure state.

EXLEVEL_NS_ELO	Meaning
0b0	The trace unit generates instruction trace for ELO in Non-secure state.
0b1	The trace unit does not generate instruction trace for ELO in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL3, bit [19]**When EL3 is implemented:**

Filter instruction trace for EL3.

EXLEVEL_S_EL3	Meaning
0b0	The trace unit generates instruction trace for EL3.
0b1	The trace unit does not generate instruction trace for EL3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL2, bit [18]**When EL2 is implemented and FEAT_SEL2 is implemented:**

Filter instruction trace for EL2 in Secure state.

EXLEVEL_S_EL2	Meaning
0b0	The trace unit generates instruction trace for EL2 in Secure state.
0b1	The trace unit does not generate instruction trace for EL2 in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL1, bit [17]**When Secure EL1 is implemented:**

Filter instruction trace for EL1 in Secure state.

EXLEVEL_S_EL1	Meaning
0b0	The trace unit generates instruction trace for EL1 in Secure state.
0b1	The trace unit does not generate instruction trace for EL1 in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_ELO, bit [16]**When Secure ELO is implemented:**

Filter instruction trace for ELO in Secure state.

EXLEVEL_S_ELO	Meaning
0b0	The trace unit generates instruction trace for ELO in Secure state.
0b1	The trace unit does not generate instruction trace for ELO in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [15:12]

Reserved, RES0.

TRCERR, bit [11]**When TRCIDR3.TRCERR == 1:**

Controls the forced tracing of System Error exceptions.

TRCERR	Meaning
0b0	Forced tracing of System Error exceptions is disabled.
0b1	Forced tracing of System Error exceptions is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TRCRESET, bit [10]

Controls the forced tracing of PE Resets.

TRCRESET	Meaning
0b0	Forced tracing of PE Resets is disabled.
0b1	Forced tracing of PE Resets is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SSSTATUS, bit [9]

ViewInst start/stop function status.

SSSTATUS	Meaning
0b0	Stopped State. The ViewInst start/stop function is in the stopped state.
0b1	Started State. The ViewInst start/stop function is in the started state.

Before software enables the trace unit, it must write to this field to set the initial state of the ViewInst start/stop function. If the ViewInst start/stop function is not used then set this field to 1. Arm recommends that the value of this field is set before each trace session begins.

If the trace unit becomes disabled while a start point or stop point is still speculative, then the value of TRCVICTLR.SSSTATUS is UNKNOWN and might represent the result of a speculative start point or stop point.

If software which is running on the PE being traced disables the trace unit, either by clearing [TRCPRGCTLR.EN](#) or locking the OS Lock, Arm recommends that a DSB and an ISB instruction are executed before disabling the trace unit to prevent any start points or stop points being speculative at the point of disabling the trace unit. This procedure assumes that all start points or stop points occur before the barrier instructions are executed. The procedure does not guarantee that there are no speculative start points or stop points when disabling, although it helps minimize the probability.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES1** if all of the following are true:
 - TRCIDR4.NUMACPAIRS == 0b0000
 - TRCIDR4.NUMPC == 0b0000
- Otherwise, access to this field is **RW**.

Bit [8]

Reserved, RES0.

EVENT_TYPE, bit [7]

When TRCIDR4.NUMRSPAIR != 0b0000:

Chooses the type of Resource Selector.

EVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCVICTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCVICTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCVICTLR.EVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [6:5]

Reserved, RES0.

Bits[4:0]**When TRCIDR4.NUMRSPAIR != 0b0000:****EVENT_SEL, bits [4:0] of bits [4:0]**

Defines the selected Resource Selector or pair of Resource Selectors. TRCVICTLR.EVENT.TYPE controls whether TRCVICTLR.EVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

When TRCIDR4.NUMRSPAIR == 0b0000:**Reserved, bits [4:0] of bits [4:0]**

This field is reserved:

- Bits [4:1] are RES0.
- Bit [0] is RES1.

Otherwise:

Reserved, RES0.

Accessing TRCVICTLR

Must be programmed.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVICTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRCVICTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVICTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVICTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVICTLR;

```

MSR TRCVICTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRCVICTLR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVICTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVICTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVICTLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

INCLUDE[<m>], bit [m], for m = 7 to 0

Include Address Range Comparator <m>.

Selects whether Address Range Comparator <m> is in use with the ViewInst include function.

Selecting no comparators for the ViewInst include function indicates that all instructions are included by default.

The ViewInst exclude function then indicates which ranges are excluded.

INCLUDE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected for the ViewInst include function.
0b1	The address range that Address Range Comparator <m> defines, is selected for the ViewInst include function.

This bit is RES0 if m >= [TRCIDR4.NUMACPAIRS](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVIIECTLR

Must be programmed if [TRCIDR4.NUMACPAIRS](#) > 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVIIECTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVIIIECTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVIIIECTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVIIIECTLR;

```

MSR TRCVIIIECTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVIIECTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVIIECTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVIIECTLR = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCVIPCSSCTLR characteristics are:

Use this to select, or read, which PE Comparator Inputs can control the ViewInst start/stop function.

AArch64 System register TRCVIPCSSCTLR bits [31:0] are architecturally mapped to External register [TRCVIPCSSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and UInt(TRCIDR4.NUMPC) > 0. Otherwise, direct accesses to TRCVIPCSSCTLR are UNDEFINED.

TRCVIPCSSCTL is a 64-bit register.

Field descriptions

6362616059585756	55	54	53	52	51	50	49	48	4746454443424140	39	38
RES0											
RES0	STOP[7]	STOP[6]	STOP[5]	STOP[4]	STOP[3]	STOP[2]	STOP[1]	STOP[0]	RES0	START[7]	START
3130292827262524	23	22	21	20	19	18	17	16	15141312111098	7	6

Bits [63:24]

Reserved, RES0.

STOP[<m>], bit [m+16], for m = 7 to 0

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of stopping trace.

STOP[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a stop resource.
0b1	The PE Comparator Input <m>, is selected as a stop resource.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMPC}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [15:8]

Reserved, RES0.

START[<m>], bit [m], for m = 7 to 0

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of starting trace.

START[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a start resource.
0b1	The PE Comparator Input <m>, is selected as a start resource.

This bit is RES0 if m >= [TRCIDR4.NUMPC](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVIPCSSCTLR

Must be programmed if [TRCIDR4.NUMPC](#) != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVIPCSSCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVIPCSSCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVIPCSSCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVIPCSSCTLR;

```

MSR TRCVIPCSSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVIPCSSCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVIPCSSCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVIPCSSCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVISSCTLR, ViewInst Start/Stop Control Register

The TRCVISSCTLR characteristics are:

Purpose

Use this to select, or read, the Single Address Comparators for the ViewInst start/stop function.

Configuration

AArch64 System register TRCVISSCTLR bits [31:0] are architecturally mapped to External register [TRCVISSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and `UInt(TRCIDR4.NUMACPAIRS) > 0`. Otherwise, direct accesses to TRCVISSCTLR are UNDEFINED.

Attributes

TRCVISSCTLR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51
STOP[15]	STOP[14]	STOP[13]	STOP[12]	STOP[11]	STOP[10]	STOP[9]	STOP[8]	STOP[7]	STOP[6]	STOP[5]	STOP[4]	STOP[3]
31	30	29	28	27	26	25	24	23	22	21	20	19

Bits [63:32]

Reserved, RES0.

STOP[<m>], bit [m+16], for m = 15 to 0

Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of stopping trace.

STOP[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected as a stop resource.
0b1	The Single Address Comparator <m>, is selected as a stop resource.

This bit is RES0 if $m \geq 2 \times \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

START[<m>], bit [m], for m = 15 to 0

Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of starting trace.

START[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected as a start resource.
0b1	The Single Address Comparator <m>, is selected as a start resource.

This bit is RES0 if $m \geq 2 \times \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVISSCTLR

Must be programmed if $\text{TRCIDR4.NUMACPAIRS} > 0b0000$.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVISSCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVISSCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVISSCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVISSCTLR;

```

MSR TRCVISSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVISSCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVISSCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVISSCTLR = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVMIDCCTLR0, Virtual Context Identifier Comparator Control Register 0

The TRCVMIDCCTLR0 characteristics are:

Purpose

Virtual Context Identifier Comparator mask values for the [TRCVMIDCVR<n>](#) registers, where n=0-3.

Configuration

AArch64 System register TRCVMIDCCTLR0 bits [31:0] are architecturally mapped to External register [TRCVMIDCCTLR0\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 0 \times 0$ and $\text{UInt}(\text{TRCIDR2.VMIDSIZE}) > 0$. Otherwise, direct accesses to TRCVMIDCCTLR0 are UNDEFINED.

Attributes

TRCVMIDCCTLR0 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	
COMP3[7]	COMP3[6]	COMP3[5]	COMP3[4]	COMP3[3]	COMP3[2]	COMP3[1]	COMP3[0]	COMP2[7]	COMP2[6]	COMP2[5]	COMP2[4]
31	30	29	28	27	26	25	24	23	22	21	

Bits [63:32]

Reserved, RES0.

COMP3[<m>], bit [m+24], for m = 7 to 0
When $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 3$:

TRCVMIDCVR3 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR3. Each bit in this field corresponds to a byte in TRCVMIDCVR3.

COMP3[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR3[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR3[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.VMIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP2[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 2:

TRCVMIDCVR2 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR2. Each bit in this field corresponds to a byte in TRCVMIDCVR2.

COMP2[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR2[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR2[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP1[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 1:

TRCVMIDCVR1 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR1. Each bit in this field corresponds to a byte in TRCVMIDCVR1.

COMP1[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR1[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR1[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP0[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 0:

TRCVMIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0.

COMP0[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCVMIDCCTLRO

If software uses the [TRCVMIDCVR<n>](#) registers, where n=0-3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCVMIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCVMIDCVR<n>](#) is not 0x00, the behavior of the Virtual Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVMIDCCTLRO

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCCTLR0;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = TRCVMIDCCTLR0;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCCTLR0;

```

MSR TRCVMIDCCTLR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0010	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDBGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCCTLR0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCCTLR0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVMIDCCTLR0 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVMIDCCTLR1, Virtual Context Identifier Comparator Control Register 1

The TRCVMIDCCTLR1 characteristics are:

Purpose

Virtual Context Identifier Comparator mask values for the [TRCVMIDCVR<n>](#) registers, where n=4-7.

Configuration

AArch64 System register TRCVMIDCCTLR1 bits [31:0] are architecturally mapped to External register [TRCVMIDCCTLR1\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented**, $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 0x4$ and $\text{UInt}(\text{TRCIDR2.VMIDSIZE}) > 0$. Otherwise, direct accesses to TRCVMIDCCTLR1 are UNDEFINED.

Attributes

TRCVMIDCCTLR1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53					
COMP7[7]	COMP7[6]	COMP7[5]	COMP7[4]	COMP7[3]	COMP7[2]	COMP7[1]	COMP7[0]	COMP6[7]	COMP6[6]	COMP6[5]	COMP6[4]	COMP6[3]	COMP6[2]	COMP6[1]	COMP6[0]
31	30	29	28	27	26	25	24	23	22	21					

Bits [63:32]

Reserved, RES0.

COMP7[<m>], bit [m+24], for m = 7 to 0 When UInt(TRCIDR4.NUMVMIDC) > 7:

TRCVMIDCVR7 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR7. Each bit in this field corresponds to a byte in TRCVMIDCVR7.

COMP7[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR7[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR7[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP6[<m>], bit [m+16], for m = 7 to 0**When UInt(TRCIDR4.NUMVMIDC) > 6:**

TRCVMIDCVR6 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR6. Each bit in this field corresponds to a byte in TRCVMIDCVR6.

COMP6[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR6[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR6[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP5[<m>], bit [m+8], for m = 7 to 0**When UInt(TRCIDR4.NUMVMIDC) > 5:**

TRCVMIDCVR5 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR5. Each bit in this field corresponds to a byte in TRCVMIDCVR5.

COMP5[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR5[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR5[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP4[<m>], bit [m], for m = 7 to 0**When UInt(TRCIDR4.NUMVMIDC) > 4:**

TRCVMIDCVR4 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR4. Each bit in this field corresponds to a byte in TRCVMIDCVR4.

COMP4[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR4[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR4[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCVMIDCCTLR1

If software uses the [TRCVMIDCVR<n>](#) registers, where n=4-7, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCVMIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCVMIDCVR<n>](#) is not 0x00, the behavior of the Virtual Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVMIDCCTLR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCCTLR1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCCTLR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVMIDCCTLR1;

```

MSR TRCVMIDCCTLR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0011	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCCTLR1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCCTLR1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVMIDCCTLR1 = X[t, 64];

```

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n>, n = 0 - 7

The TRCVMIDCVR<n> characteristics are:

Purpose

Contains the Virtual Context Identifier Comparator value.

Configuration

AArch64 System register TRCVMIDCVR<n> bits [63:0] are architecturally mapped to External register [TRCVMIDCVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_SR is implemented** and $UInt(TRCIDR4.NUMVMIDC) > n$. Otherwise, direct accesses to TRCVMIDCVR<n> are UNDEFINED.

Attributes

TRCVMIDCVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																VALUE															
																VALUE															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VALUE, bits [63:0]

Virtual context identifier value. The width of this field is indicated by [TRCIDR2.VMIDSIZE](#). Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Virtual context identifier is zero until the PE updates the Virtual context identifier .

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVMIDCVR<n>

Must be programmed if any of the following are true:

- [TRCRSCTLR<a>](#).GROUP == 0b0111 and [TRCRSCTLR<a>](#).VMID[n] == 1.
- [TRCACATR<a>](#).CONTEXTTYPE == 0b10 or 0b11 and [TRCACATR<a>](#).CONTEXT == n.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRCVMIDCVR<m> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:0b0	0b001

```

integer m = UInt(CRm<3:1>);

if m >= NUM_TRACE_VIRTUAL_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCVR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRCVMIDCVR[m];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVMIDCVR[m];

```

MSR TRCVMIDCVR<m>, <Xt> ; Where m = 0-7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:0b0	0b001

```

integer m = UInt(CRm<3:1>);

if m >= NUM_TRACE_VIRTUAL_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.TTA == '1' then
            UNDEFINED;
        elsif CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCVR[m] = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRCVMIDCVR[m] = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:2]

Reserved, RES0.

E1TRE, bit [1]

EL1 Trace Enable.

E1TRE	Meaning
0b0	Trace is prohibited at EL1.
0b1	Trace is allowed at EL1.

This field is ignored if SelfHostedTraceEnabled() == FALSE.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

E0TRE, bit [0]

EL0 Trace Enable.

E0TRE	Meaning
0b0	Trace is prohibited at EL0.
0b1	Trace is allowed at EL0.

This field is ignored if any of the following are true:

- SelfHostedTraceEnabled() == FALSE.
- EL2 is implemented and enabled in the current Security state and [HCR_EL2.TGE](#) == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRFCR_EL1

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRFCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x880];
    else
        X[t, 64] = TRFCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = TRFCR_EL2;
    else
        X[t, 64] = TRFCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRFCR_EL1;

```

MSR TRFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRFCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x880] = X[t, 64];
    else
        TRFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        TRFCR_EL2 = X[t, 64];
    else
        TRFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRFCR_EL1 = X[t, 64];

```

MRS <Xt>, TRFCR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x880];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = TRFCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = TRFCR_EL1;
    else
        UNDEFINED;

```

MSR TRFCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x880] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            TRFCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TRFCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

- On a Warm reset, this field resets to 0.

Bit [4]

Reserved, RES0.

CX, bit [3]

[CONTEXTIDR_EL2](#) and VMID trace enable.

CX	Meaning
0b0	CONTEXTIDR_EL2 and VMID trace prohibited.
0b1	CONTEXTIDR_EL2 and VMID trace allowed.

This field is ignored if `SelfHostedTraceEnabled() == FALSE`.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Bit [2]

Reserved, RES0.

E2TRE, bit [1]

EL2 Trace Enable.

E2TRE	Meaning
0b0	Trace is prohibited at EL2.
0b1	Trace is allowed at EL2.

This field is ignored if `SelfHostedTraceEnabled() == FALSE`.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

EOHTRE, bit [0]

EL0 Trace Enable.

EOHTRE	Meaning
0b0	Trace is prohibited at EL0 when HCR_EL2.TGE == 1.
0b1	Trace is allowed at EL0 when HCR_EL2.TGE == 1.

This field is ignored if any of the following are true:

- `SelfHostedTraceEnabled() == FALSE`.
- EL2 is disabled in the current security state.
- [HCR_EL2.TGE](#) == 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRFCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TRFCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRFCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRFCR_EL2;

```

MSR TRFCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRFCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRFCR_EL2 = X[t, 64];

```

MRS <Xt>, TRFCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x880];
    else
        X[t, 64] = TRFCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = TRFCR_EL2;
    else
        X[t, 64] = TRFCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TRFCR_EL1;

```

MSR TRFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HDFGWTR_EL2.TRFCR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x880] = X[t, 64];
    else
        TRFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TTRF == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TTRF == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        TRFCR_EL2 = X[t, 64];
    else
        TRFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    TRFCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TTBR0_EL1 characteristics are:

Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the EL1&0 translation regime, and other information for this translation regime.

Configuration

AArch64 System register TTBR0_EL1 bits [63:0] are architecturally mapped to AArch32 System register [TTBR0\[63:0\]](#).

TTBR0_EL1 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Attributes

TTBR0_EL1 is a 64-bit register.

- 128-bit register when FEAT_D128 is implemented and TCR2_EL1.D128 == 1
- 64-bit register when FEAT_D128 is not implemented or TCR2_EL1.D128 == 0

Field descriptions

When FEAT_D128 is implemented and TCR2_EL1.D128 == 1:

12712612512412312212112011911811711611511411311211110109108107106105104103102101100999897																																	
RES0																RES0																	
RES0								BADDR[50:43]												RES0													
ASID																BADDR[42:0]																	
BADDR[42:0]																														RES0		SKL	
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																BADDR[47:1]																	
BADDR[47:1]																														CnP			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [127:88]

Reserved, RES0.

BADDR, bits [87:80, 47:5]

Translation table base address:

- Bits A[55:x] of the stage 1 translation table base address bits are in register bits[87:80, 47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on LOG2(StartTableSize), as described in VMSAv9-128. The smallest permitted value of x is 5.

The BADDR field is split as follows:

- BADDR[50:43] is TTBR0_EL1[87:80].
- BADDR[42:0] is TTBR0_EL1[47:5].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [79:64]

Reserved, RES0.

ASID, bits [63:48]

An ASID for the translation table base address. The [TCR_EL1.A1](#) field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level associated with translation table walks using [TTBR0_EL1](#).

This determines the number of levels to be skipped from the regular start level of the Stage 1 EL1&0 translation table walks using [TTBR0_EL1](#).

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

When FEAT_TTCNP is implemented:

Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR0_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR0_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> The value of TTBR0_EL1.CnP on those other PEs. The value of the current ASID. If EL2 is implemented and enabled in the current Security state, the value of the current VMID.
0b1	<p>The translation table entries pointed to by TTBR0_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR0_EL1. The translation tables relate to the same translation regime. The ASID is the same as the current ASID. If EL2 is implemented and enabled in the current Security state, the value of the current VMID.

This bit is permitted to be cached in a TLB.

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

Note

If the value of the TTBR0_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNPREDICTABLE, see 'CONstrained UNPREDICTABLE behaviors due to caching of control or data values'.

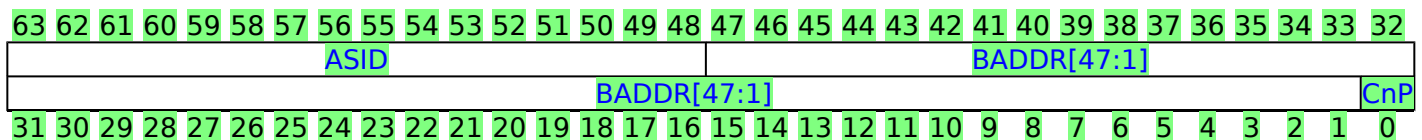
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When FEAT_D128 is not implemented or TCR2_EL1.D128 == 0:



ASID, bits [63:48]

An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BADDR[47:1], bits [47:1]

Translation table base address:

- Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR_EL1.T0SZ](#), the translation stage, and the translation granule size.

Note

A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.

If the value of [TCR_EL1.IPS](#) is not 0b110, then:

- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs, then bits A[51:48] of the stage 1 translation table base address are 0b0000.

If FEAT_LPA is implemented and the value of [TCR_EL1.IPS](#) is 0b110, then:

- Bits A[51:48] of the stage 1 translation table base address bits are in register bits[5:2].
- Register bit[1] is RES0.
- When x>6, register bits[(x-1):6] are RES0.

Note

[TCR_EL1.IPS](#)==0b110 is permitted when:

- FEAT_LPA is implemented and the 64KB translation granule is used.
- FEAT_LPA2 is implemented and the 4KB or 16KB translation granule is used.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [TCR_EL1.IPS](#) is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation supports a 56 bit PA size, bits A[55:52] of the stage 1 translation table base address are zero.

If any register bit[47:1] that is defined as RES0 has the value 1 when a translation table walk is done using TTBR0_EL1, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits A[(x-1):0] of the stage 1 translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR0_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR0_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> • The value of TTBR0_EL1.CnP on those other PEs. • The value of the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID.
0b1	<p>The translation table entries pointed to by TTBR0_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> • The translation table entries are pointed to by TTBR0_EL1. • The translation tables relate to the same translation regime. • The ASID is the same as the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID.

This bit is permitted to be cached in a TLB.

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

Note

If the value of the TTBR0_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TTBR0_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TTBR0_EL1 or TTBR0_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TTBR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x200];
    else
        X[t, 64] = TTBR0_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR0_EL2<63:0>;
    else
        X[t, 64] = TTBR0_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR0_EL1<63:0>;

```

MSR TTBR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x200] = X[t, 64];
    else
        TTBR0_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR0_EL2<63:0> = X[t, 64];
    else
        TTBR0_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    TTBR0_EL1<63:0> = X[t, 64];

```

MRS <Xt>, TTBR0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x200];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR0_EL1<63:0>;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR0_EL1<63:0>;
    else
        UNDEFINED;

```

MSR TTBR0_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x200] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR0_EL1<63:0> = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TTBR0_EL1<63:0> = X[t, 64];
    else
        UNDEFINED;

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (X[t + 1, 64], X[t, 64]) = (NVMem[0x208], NVMem[0x200]);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (X[t + 1, 64], X[t, 64]) = (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
    elsif PSTATE.EL == EL3 then
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR0_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (NVMem[0x208], NVMem[0x200]) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        (X[t + 1, 64], X[t, 64]) = (NVMem[0x208], NVMem[0x200]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
    else
        UNDEFINED;

```

When FEAT_D128 is implemented

MSRR TTBR0_EL12, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        (NVMem[0x208], NVMem[0x200]) = (X[t + 1, 64], X[t, 64]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TTBR0_EL2, Translation Table Base Register 0 (EL2)

The TTBR0_EL2 characteristics are:

Purpose

When [HCR_EL2.E2H](#) is 0, holds the base address of the translation table for the initial lookup for stage 1 of an address translation in the EL2 translation regime, and other information for this translation regime.

When [HCR_EL2.E2H](#) is 1, holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the EL2&0 translation regime, and other information for this translation regime.

Configuration

AArch64 System register TTBR0_EL2 bits [47:1] are architecturally mapped to AArch32 System register [HTTBR\[47:1\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

TTBR0_EL2 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

Attributes

TTBR0_EL2 is a: ~~64-bit register.~~

- 128-bit register when [FEAT_D128](#) is implemented, [TCR2_EL2.D128](#) == 1 and [HCR_EL2.E2H](#) == 1
- 64-bit register when [FEAT_D128](#) is not implemented or [TCR2_EL2.D128](#) == 0

Field descriptions

When [FEAT_D128](#) is implemented, [TCR2_EL2.D128](#) == 1 and [HCR_EL2.E2H](#) == 1:

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97			
RES0																																	
RES0								BADDR[55:5][50:43]																RES0									
ASID																BADDR[55:5][42:0]																	
BADDR[55:5][42:0]																																RES0	SKL
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
ASID																BADDR[47:1]																	
BADDR[47:1]																															CnP		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [127:88]

Reserved, RES0.

BADDR[55:5], bits [87:80, 47:5]

Translation table base address:

- Bits A[55:x] of the stage 1 translation table base address bits are in register bits[87:80, 47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on $\text{LOG2}(\text{StartTableSize})$, as described in VMSAv9-128. The smallest permitted value of x is 5.

The BADDR[55:5] field is split as follows:

- BADDR[55:5][50:43] is TTBR0_EL2[87:80].
- BADDR[55:5][42:0] is TTBR0_EL2[47:5].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [79:64]

Reserved, RES0.

ASID, bits [63:48]

When FEAT_VHE is implemented:

When HCR_EL2.E2H is 1, it holds an ASID for the translation table base address. The TCR_EL2.A1 field selects either TTBR0_EL2.ASID or TTBR1_EL2.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level associated with translation table walks using TTBR0_EL2.

This determines the number of levels to be skipped from the regular start level of the Stage 1 EL2&0 translation table walks using TTBR0_EL2.

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

When FEAT_TTCNP is implemented:

Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL2.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR0_EL2 for the current translation regime, and ASID if applicable, are permitted to differ from corresponding entries for TTBR0_EL2 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> The value of TTBR0_EL2.CnP on those other PEs. When the current translation regime is the EL2&0 regime, the value of the current ASID.
0b1	<p>The translation table entries pointed to by TTBR0_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL2.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR0_EL2. The translation tables relate to the same translation regime. If that translation regime is the EL2&0 regime, the ASID is the same as the current ASID.

This bit is permitted to be cached in a TLB.

Note

If the value of the TTBR0_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are **CONSTRAINED UNPREDICTABLE**, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

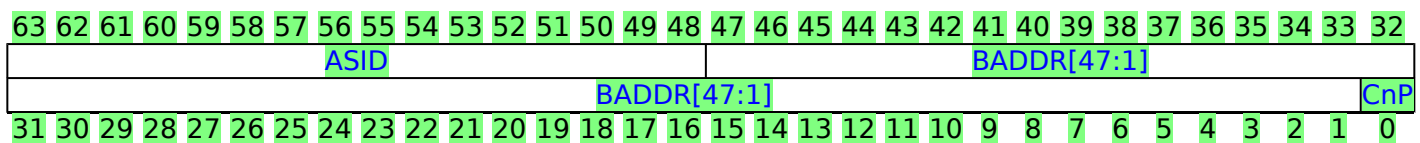
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When FEAT_D128 is not implemented or TCR2_EL2.D128 == 0:



ASID, bits [63:48]

When FEAT_VHE is implemented:

When [HCR_EL2.E2H](#) is 0, this field is RES0.

When [HCR_EL2.E2H](#) is 1, it holds an ASID for the translation table base address. The [TCR_EL2.A1](#) field selects either TTBR0_EL2.ASID or TTBR1_EL2.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BADDR[47:1], bits [47:1]

Translation table base address:

- Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR_EL2.T0SZ](#), the translation stage, and the translation granule size.

Note

A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.

If the value of [TCR_EL2.{I}PS](#) is not 0b110, then:

- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs, then bits A[51:48] of the stage 1 translation table base address are 0b0000.

If FEAT_LPA is implemented and the value of [TCR_EL2.{I}PS](#) is 0b110, then:

- Bits A[51:48] of the stage 1 translation table base address bits are in register bits[5:2].
- Register bit[1] is RES0.
- When x>6, register bits[(x-1):6] are RES0.

Note

The OA size specified by [TCR_EL2.{I}PS](#) is determined as follows:

- The value of [TCR_EL2.PS](#) when the value of [HCR_EL2.E2H](#) is 0.
- The value of [TCR_EL2.IPS](#) when the value of [HCR_EL2.E2H](#) is 1.

[TCR_EL2.{I}PS](#)==0b110 is permitted when:

- FEAT_LPA is implemented and the 64KB translation granule is used.
- FEAT_LPA2 is implemented and the 4KB or 16KB translation granule is used.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [TCR_EL2.{I}PS](#) is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation supports a 56 bit PA size, bits A[55:52] of the stage 1 translation table base address are zero.

If any register bit[47:1] that is defined as RES0 has the value 1 when a translation table walk is done using TTBR0_EL2, then the translation table base address might be misaligned, with effects that are CONstrained UNPREDICTABLE, and must be one of the following:

- Bits A[(x-1):0] of the stage 1 translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL2.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR0_EL2 for the current translation regime, and ASID if applicable, are permitted to differ from corresponding entries for TTBR0_EL2 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> The value of TTBR0_EL2.CnP on those other PEs. When the current translation regime is the EL2&0 regime, the value of the current ASID.
0b1	<p>The translation table entries pointed to by TTBR0_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL2.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR0_EL2. The translation tables relate to the same translation regime. If that translation regime is the EL2&0 regime, the ASID is the same as the current ASID.

This bit is permitted to be cached in a TLB.

Note

If the value of the TTBR0_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TTBR0_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TTBR0_EL2 or TTBR0_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TTBR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TTBR0_EL2<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR0_EL2<63:0>;

```

MSR TTBR0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    TTBR0_EL2<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    TTBR0_EL2<63:0> = X[t, 64];

```

MRS <Xt>, TTBR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x200];
    else
        X[t, 64] = TTBR0_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR0_EL2<63:0>;
    else
        X[t, 64] = TTBR0_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR0_EL1<63:0>;

```

MSR TTBR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGWTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x200] = X[t, 64];
    else
        TTBR0_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            TTBR0_EL2<63:0> = X[t, 64];
        else
            TTBR0_EL1<63:0> = X[t, 64];
    elsif PSTATE.EL == EL3 then
        TTBR0_EL1<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>);
    elsif PSTATE.EL == EL3 then
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR0_EL2, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
elsif PSTATE.EL == EL3 then
    (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        (X[t + 1, 64], X[t, 64]) = (NVMem[0x208], NVMem[0x200]);
    else
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif HCR_EL2.E2H == '1' then
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>);
    else
        (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR0_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (NVMem[0x208], NVMem[0x200]) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (TTBR0_EL2<127:64>, TTBR0_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (TTBR0_EL1<127:64>, TTBR0_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TTBR0_EL3, Translation Table Base Register 0 (EL3)

The TTBR0_EL3 characteristics are:

Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of an address translation in the EL3 translation regime, and other information for this translation regime.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to TTBR0_EL3 are UNDEFINED.

Attributes

TTBR0_EL3 is a 64-bit register.

Field descriptions

When FEAT_D128 is implemented and TCR_EL3.D128 == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0								BADDRBADDR[47:1]																								
BADDRBADDR[47:1]																RES0CnP				SKL		CnP										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:56]

Reserved, RES0.

BADDR, bits [55:5]

- Bits A[55:x] of the stage 1 translation table base address bits are in register bits[55:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on LOG2(StartTableSize), as described in VMSAv9-128. The smallest permitted value of x is 5.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level associated with translation table walks using TTBR0_EL3.

This determines the number of levels to be skipped from the regular start level of the Stage 1 EL3 translation table walks using TTBR0_EL3.

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

Common not Private, for stage 2 of the Secure EL1&0 translation regime. In an implementation that includes FEAT_TTCNP, indicates whether each entry that is pointed to by VSTTBR_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by VSTTBR_EL2 are permitted to differ from the entries for VSTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
0b1	The translation table entries pointed to by VSTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

This bit is permitted to be cached in a TLB.

Note

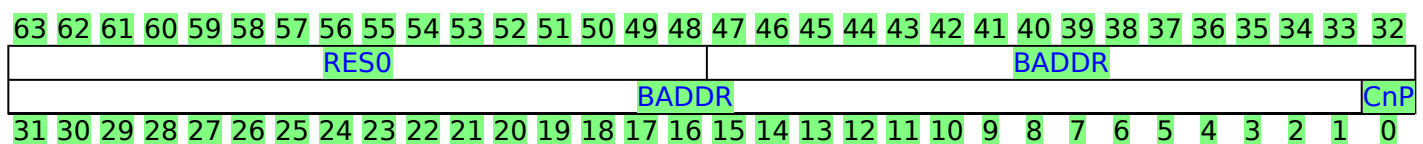
If the value of VSTTBR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VSTTBR_EL2s do not point to the same translation table entries when using the current VMID, then the results of translations using VSTTBR_EL2 are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is not implemented or TCR_EL3.D128 == 0:



Bits [63:48]

Reserved, RES0.

BADDR, [47:1], bits [47:1]

Translation table base address:

- Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR_EL3.T0SZ](#), the translation stage, and the translation granule size.

Note

A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.

If the value of [TCR_EL3.PS](#) is not 0b110, then:

- Register bits[($x-1$):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs, then bits A[51:48] of the stage 1 translation table base address are 0b0000.

If FEAT_LPA is implemented and the value of [TCR_EL3.PS](#) is 0b110, then:

- Bits A[51:48] of the stage 1 translation table base address bits are in register bits[5:2].
 - Register bit[1] is RES0.
 - When $x > 6$, register bits[($x-1$):6] are RES0.
-

Note

[TCR_EL3.PS](#)==0b110 is permitted when:

- FEAT_LPA is implemented and the 64KB translation granule is used.
- FEAT_LPA2 is implemented and the 4KB or 16KB translation granule is used.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [TCR_EL3.PS](#) is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation supports a 56 bit PA size, bits A[55:52] of the stage 1 translation table base address are zero.

If any register bit[47:1] that is defined as RES0 has the value 1 when a translation table walk is done using TTBR0_EL3, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits A[($x-1$):0] of the stage 1 translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

When FEAT_TTCNP is implemented:

Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL3 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL3.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by TTBR0_EL3, for the current translation regime, are permitted to differ from corresponding entries for TTBR0_EL3 for other PEs in the Inner Shareable domain. This is not affected by the value of TTBR0_EL3.CnP on those other PEs.
0b1	The translation table entries pointed to by TTBR0_EL3 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL3.CnP is 1 and the translation table entries are pointed to by TTBR0_EL3.

This bit is permitted to be cached in a TLB.

Note

If the value of the TTBR0_EL3.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL3s do not point to the same translation table entries the results of translations using TTBR0_EL3 are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TTBR0_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TTBR0_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR0_EL3;

```

MSR TTBR0_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    TTBR0_EL3 = X[t, 64];
```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TTBR1_EL1, Translation Table Base Register 1 (EL1)

The TTBR1_EL1 characteristics are:

Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the higher VA range in the EL1&0 stage 1 translation regime, and other information for this translation regime.

Configuration

AArch64 System register TTBR1_EL1 bits [63:0] are architecturally mapped to AArch32 System register [TTBR1\[63:0\]](#).

TTBR1_EL1 is a 128-bit register that can also be accessed as a 64-bit value. If it is accessed as a 64-bit register, accesses read and write bits [63:0] and do not modify bits [127:64].

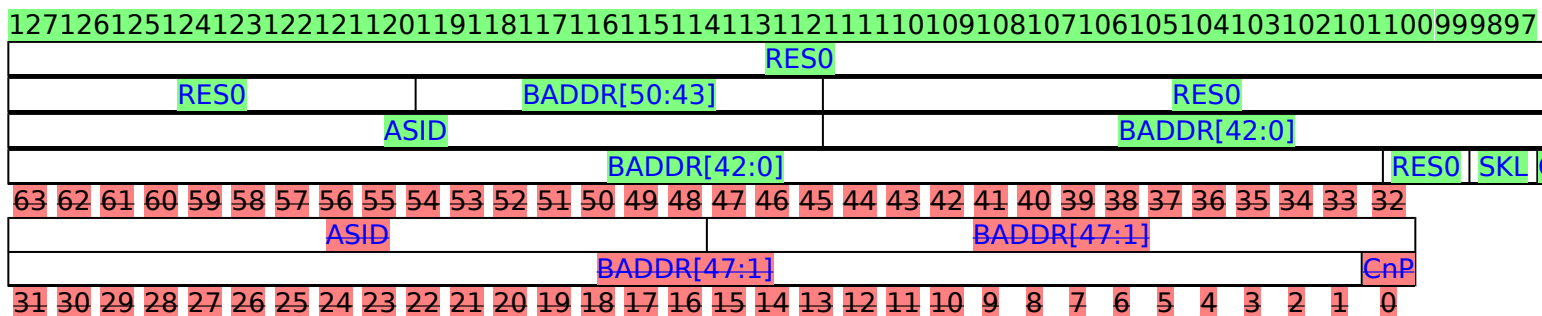
Attributes

TTBR1_EL1 is a: ~~64-bit register.~~

- 128-bit register when FEAT_D128 is implemented and TCR2_EL1.D128 == 1
- 64-bit register when FEAT_D128 is not implemented or TCR2_EL1.D128 == 0

Field descriptions

When FEAT_D128 is implemented and TCR2_EL1.D128 == 1:



Bits [127:88]

Reserved, RES0.

BADDR, bits [87:80, 47:5]

Translation table base address:

- Bits A[55:x] of the stage 1 translation table base address bits are in register bits[87:80, 47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on LOG2(StartTableSize), as described in VMSAv9-128. The smallest permitted value of x is 5.

The BADDR field is split as follows:

- BADDR[50:43] is TTBR1_EL1[87:80].
- BADDR[42:0] is TTBR1_EL1[47:5].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [79:64]

Reserved, RES0.

ASID, bits [63:48]

An ASID for the translation table base address. The [TCR_EL1.A1](#) field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level associated with translation table walks using [TTBR1_EL1](#).

This determines the number of levels to be skipped from the regular start level of the Stage 1 EL1&0 translation table walks using [TTBR1_EL1](#).

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

When FEAT_TTCNP is implemented:

Common not Private. This bit indicates whether each entry that is pointed to by TBR1_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR1_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR1_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> The value of TTBR1_EL1.CnP on those other PEs. The value of the current ASID. If EL2 is implemented and enabled in the current Security state, the value of the current VMID.
0b1	<p>The translation table entries pointed to by TTBR1_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR1_EL1. The translation tables relate to the same translation regime. The ASID is the same as the current ASID. If EL2 is implemented and enabled in the current Security state, the value of the current VMID.

This bit is permitted to be cached in a TLB.

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

Note

If the value of the TTBR1_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNPREDICTABLE, see 'CONstrained UNPREDICTABLE behaviors due to caching of control or data values'.

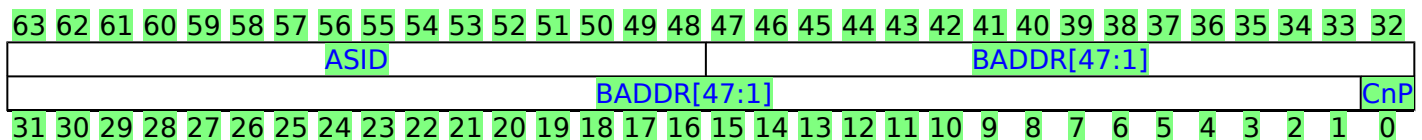
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When FEAT_D128 is not implemented or TCR2_EL1.D128 == 0:



ASID, bits [63:48]

An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BADDR[47:1], bits [47:1]

Translation table base address:

- Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR_EL1.T1SZ](#), the translation stage, and the translation granule size.

Note

A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.

If the value of [TCR_EL1.IPS](#) is not 0b110, then:

- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs, then bits A[51:48] of the stage 1 translation table base address are 0b0000.

If FEAT_LPA is implemented and the value of [TCR_EL1.IPS](#) is 0b110, then:

- Bits A[51:48] of the stage 1 translation table base address bits are in register bits[5:2].
- Register bit[1] is RES0.
- When x>6, register bits[(x-1):6] are RES0.

Note

[TCR_EL1.IPS](#)==0b110 is permitted when:

- FEAT_LPA is implemented and the 64KB translation granule is used.
- FEAT_LPA2 is implemented and the 4KB or 16KB translation granule is used.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [TCR_EL1.IPS](#) is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated.

When the value of [ID_AA64MMFR0_EL1.PARange](#) indicates that the implementation supports a 56 bit PA size, bits A[55:52] of the stage 1 translation table base address are zero.

If any register bit[47:1] that is defined as RES0 has the value 1 when a translation table walk is done using TTBR1_EL1, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits A[(x-1):0] of the stage 1 translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by TBR1_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1.

CnP	Meaning
0b0	<p>The translation table entries pointed to by TTBR1_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR1_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> • The value of TTBR1_EL1.CnP on those other PEs. • The value of the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID.
0b1	<p>The translation table entries pointed to by TTBR1_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> • The translation table entries are pointed to by TTBR1_EL1. • The translation tables relate to the same translation regime. • The ASID is the same as the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID.

This bit is permitted to be cached in a TLB.

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

Note

If the value of the TTBR1_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TTBR1_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TTBR1_EL1 or TTBR1_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TTBR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x210];
    else
        X[t, 64] = TTBR1_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR1_EL2<63:0>;
    else
        X[t, 64] = TTBR1_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR1_EL1<63:0>;

```

MSR TTBR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x210] = X[t, 64];
    else
        TTBR1_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR1_EL2<63:0> = X[t, 64];
    else
        TTBR1_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    TTBR1_EL1<63:0> = X[t, 64];

```

MRS <Xt>, TTBR1_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x210];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR1_EL1<63:0>;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR1_EL1<63:0>;
    else
        UNDEFINED;

```

MSR TTBR1_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x210] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR1_EL1<63:0> = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TTBR1_EL1<63:0> = X[t, 64];
    else
        UNDEFINED;

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (X[t + 1, 64], X[t, 64]) = (NVMem[0x218], NVMem[0x210]);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (X[t + 1, 64], X[t, 64]) = (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
    elsif PSTATE.EL == EL3 then
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR1_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() || HCRX_EL2.D128En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (NVMem[0x218], NVMem[0x210]) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR1_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        (X[t + 1, 64], X[t, 64]) = (NVMem[0x218], NVMem[0x210]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
    else
        UNDEFINED;

```

When FEAT_D128 is implemented

MSRR TTBR1_EL12, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        (NVMem[0x218], NVMem[0x210]) = (X[t + 1, 64], X[t, 64]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

BADDR, bits [87:80, 47:5]

Translation table base address:

- Bits A[55:x] of the stage 1 translation table base address bits are in register bits[87:80, 47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on $\text{LOG2}(\text{StartTableSize})$, as described in VMSAv9-128. The smallest permitted value of x is 5.

The BADDR field is split as follows:

- BADDR[50:43] is TTBR1_EL2[87:80].
- BADDR[42:0] is TTBR1_EL2[47:5].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [79:64]

Reserved, RES0.

ASID, bits [63:48]

An ASID for the translation table base address. The TCR_EL2.A1 field selects either TTBR0_EL2.ASID or TTBR1_EL2.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level associated with translation table walks using TTBR1_EL2.

This determines the number of levels to be skipped from the regular start level of the Stage 1 EL2&0 translation table walks using TTBR1_EL2.

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by TBR1_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by TTBR1_EL2 for the current ASID are permitted to differ from corresponding entries for TTBR1_EL2 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> The value of TTBR1_EL2.CnP on those other PEs. The value of the current ASID.
0b1	The translation table entries pointed to by TTBR1_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL2.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR1_EL2. The ASID is the same as the current ASID.

This bit is permitted to be cached in a TLB.

Note

- TTBR1_EL2 is accessible only when the value of HCR_EL2.E2H is 1, meaning the current translation regime is the EL2&0 regime.
- If the value of the TTBR1_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

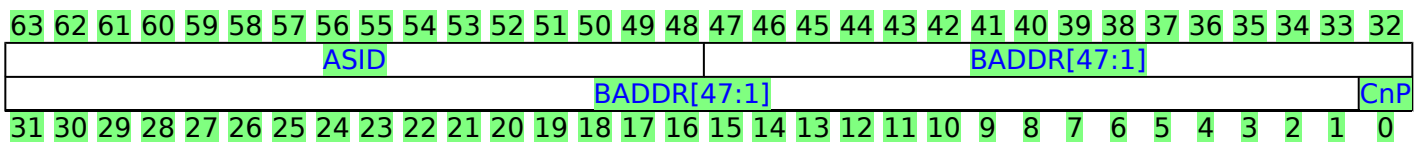
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When FEAT_D128 is not implemented or TCR2_EL2.D128 == 0:



ASID, bits [63:48]

An ASID for the translation table base address. The TCR_EL2.A1 field selects either TTBR0_EL2.ASID or TTBR1_EL2.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BADDR[47:1], bits [47:1]

Translation table base address:

- Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x].
- Bits A[(x-1):0] of the stage 1 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of TCR_EL2.T1SZ, the translation stage, and the translation granule size.

Note

A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.

If the value of [TCR_EL2](#).{I}PS is not 0b110, then:

- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs, then bits A[51:48] of the stage 1 translation table base address are 0b0000.

If FEAT_LPA is implemented and the value of [TCR_EL2](#).{I}PS is 0b110, then:

- Bits A[51:48] of the stage 1 translation table base address bits are in register bits[5:2].
- Register bit[1] is RES0.
- When x>6, register bits[(x-1):6] are RES0.

Note

The OA size specified by [TCR_EL2](#).{I}PS is determined as follows:

- The value of [TCR_EL2](#).PS when the value of [HCR_EL2](#).E2H is 0.
- The value of [TCR_EL2](#).IPS when the value of [HCR_EL2](#).E2H is 1.

[TCR_EL2](#).{I}PS==0b110 is permitted when:

- FEAT_LPA is implemented and the 64KB translation granule is used.
- FEAT_LPA2 is implemented and the 4KB or 16KB translation granule is used.

When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [TCR_EL2](#).{I}PS is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated.

When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation supports a 56 bit PA size, bits A[55:52] of the stage 1 translation table base address are zero.

If any register bit[47:1] that is defined as RES0 has the value 1 when a translation table walk is done using TTBR1_EL2, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits A[(x-1):0] of the stage 1 translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by TBR1_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by TTBR1_EL2 for the current ASID are permitted to differ from corresponding entries for TTBR1_EL2 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> The value of TTBR1_EL2.CnP on those other PEs. The value of the current ASID.
0b1	The translation table entries pointed to by TTBR1_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL2.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> The translation table entries are pointed to by TTBR1_EL2. The ASID is the same as the current ASID.

This bit is permitted to be cached in a TLB.

Note

- TTBR1_EL2 is accessible only when the value of [HCR_EL2.E2H](#) is 1, meaning the current translation regime is the EL2&0 regime.
- If the value of the TTBR1_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TTBR1_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TTBR1_EL2 or TTBR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, TTBR1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TTBR1_EL2<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR1_EL2<63:0>;

```

MSR TTBR1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    TTBR1_EL2<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    TTBR1_EL2<63:0> = X[t, 64];

```

MRS <Xt>, TTBR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x210];
    else
        X[t, 64] = TTBR1_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = TTBR1_EL2<63:0>;
    else
        X[t, 64] = TTBR1_EL1<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = TTBR1_EL1<63:0>;

```

MSR TTBR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x210] = X[t, 64];
    else
        TTBR1_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR1_EL2<63:0> = X[t, 64];
    else
        TTBR1_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    TTBR1_EL1<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR1_EL2, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
elsif PSTATE.EL == EL3 then
    (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, TTBR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        (X[t + 1, 64], X[t, 64]) = (NVMem[0x218], NVMem[0x210]);
    else
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    elsif HCR_EL2.E2H == '1' then
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>);
    else
        (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>);

```

When FEAT_D128 is implemented

MSRR TTBR1_EL1, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.TTBR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            (NVMem[0x218], NVMem[0x210]) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x14);
        elsif HCR_EL2.E2H == '1' then
            (TTBR1_EL2<127:64>, TTBR1_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
        else
            (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);
    elsif PSTATE.EL == EL3 then
        (TTBR1_EL1<127:64>, TTBR1_EL1<63:0>) = (X[t + 1, 64], X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VBAR_EL1, Vector Base Address Register (EL1)

The VBAR_EL1 characteristics are:

Purpose

Holds the vector base address for any exception that is taken to EL1.

Configuration

AArch64 System register VBAR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [VBAR\[31:0\]](#).

Attributes

VBAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Vector Base Address																															
Vector Base Address																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL1.

-
- If tagged addresses are not being used, bits [63:48:52] of VBAR_EL1 must be the same or else the use of the vector address will result in a recursive exception.
-

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [10:0]

Reserved, RES0.

Accessing VBAR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic VBAR_EL1 or VBAR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VBAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.VBAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x250];
    else
        X[t, 64] = VBAR_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = VBAR_EL2;
    else
        X[t, 64] = VBAR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = VBAR_EL1;

```

MSR VBAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.VBAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x250] = X[t, 64];
    else
        VBAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        VBAR_EL2 = X[t, 64];
    else
        VBAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VBAR_EL1 = X[t, 64];

```

MRS <Xt>, VBAR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x250];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = VBAR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        X[t, 64] = VBAR_EL1;
    else
        UNDEFINED;

```

MSR VBAR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x250] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        VBAR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        VBAR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VBAR_EL2, Vector Base Address Register (EL2)

The VBAR_EL2 characteristics are:

Purpose

Holds the vector base address for any exception that is taken to EL2.

Configuration

AArch64 System register VBAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HVBAR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VBAR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Vector Base Address																															
Vector Base Address																RES0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL2.

Note

If [FEAT_LVA3](#) is implemented:

- If [HCR_EL2.E2H](#) == 0b1:
 - If tagged addresses are not being used, bits [63:52] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are **not** being used, bits [63:52] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
- If [HCR_EL2.E2H](#) == 0b0:
 - If tagged addresses are not being used, bits [63:52] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are **not** being used, bits [63:52] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

Otherwise :

If [FEAT_LVA](#) is implemented:

- If [HCR_EL2.E2H](#) == 0b1:

-
- If tagged addresses are being used, bits [55:52] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are not being used, bits [63:52] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
 - If `HCR_EL2.E2H == 0b0`:
 - If tagged addresses are being used, bits [55:52] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are not being used, bits [63:52] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

If FEAT_LVA is not implemented:

- If `HCR_EL2.E2H == 0b1`:
 - If tagged addresses are being used, bits [55:48] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are not being used, bits [63:48] of VBAR_EL2 must be the same or else the use of the vector address will result in a recursive exception.
 - If `HCR_EL2.E2H == 0b0`:
 - If tagged addresses are being used, bits [55:48] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
 - If tagged addresses are not being used, bits [63:48] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
-

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [10:0]

Reserved, RES0.

Accessing VBAR_EL2

When `HCR_EL2.E2H` is 1, without explicit synchronization, access from EL2 using the mnemonic VBAR_EL2 or VBAR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VBAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VBAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VBAR_EL2;

```

MSR VBAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VBAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VBAR_EL2 = X[t, 64];

```

MRS <Xt>, VBAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
    && HFGTR_EL2.VBAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x250];
    else
        X[t, 64] = VBAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = VBAR_EL2;
    else
        X[t, 64] = VBAR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VBAR_EL1;

```

MSR VBAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1')
&& HFGWTR_EL2.VBAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x250] = X[t, 64];
    else
        VBAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        VBAR_EL2 = X[t, 64];
    else
        VBAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VBAR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VBAR_EL3, Vector Base Address Register (EL3)

The VBAR_EL3 characteristics are:

Purpose

Holds the vector base address for any exception that is taken to EL3.

Configuration

This register is present only when EL3 is implemented. Otherwise, direct accesses to VBAR_EL3 are UNDEFINED.

Attributes

VBAR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Vector Base Address																															
Vector Base Address										RES0																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL3.

- If tagged addresses are not being used, bits [63:4852] of VBAR_EL3 must be 0 or else the use of the vector address will result in a recursive exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [10:0]

Reserved, RES0.

Accessing VBAR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VBAR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VBAR_EL3;
```

```
MSR VBAR_EL3, <Xt>
```

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    VBAR EL3 = X[t, 64];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

VMECID_A_EL2, Alternate MECID for EL1&0 stage 2 translation regime

The VMECID_A_EL2 characteristics are:

Purpose

Alternate MECID for EL1&0 stage 2 translation regime.

Configuration

This register is present only when FEAT_MEC is implemented, IsCurrentSecurityState(SS_Realm) and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to VMECID_A_EL2 are UNDEFINED.

Attributes

VMECID_A_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																MECID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VMECID_A_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VMECID_A_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1001	0b001

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x308];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VMECID_A_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VMECID_A_EL2;
```

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

no old file

htmldiff from-

(new)

VMECID_P_EL2, Primary MECID for EL1&0 stage 2 translation regime

The VMECID_P_EL2 characteristics are:

Purpose

Primary MECID for EL1&0 stage 2 translation regime.

Configuration

This register is present only when FEAT_MEC is implemented, IsCurrentSecurityState(SS_Realm) and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to VMECID_P_EL2 are UNDEFINED.

Attributes

VMECID_P_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																RES0															
RES0																MECID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:16]

Reserved, RES0.

MECID, bits [15:0]

If MECIDWidth is less than 16, bits[15:MECIDWidth] are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VMECID_P_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VMECID_P_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b1001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x300];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VMECID_P_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VMECID_P_EL2;
```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

VMPIDR_EL2, Virtualization Multiprocessor ID Register

The VMPIDR_EL2 characteristics are:

Purpose

Holds the value of the Virtualization Multiprocessor ID. This is the value returned by EL1 reads of [MPIDR_EL1](#).

Configuration

AArch64 System register VMPIDR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [VMPIDR\[31:0\]](#).

If EL2 is not implemented, reads of this register return the value of the [MPIDR_EL1](#) and writes to the register are ignored.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VMPIDR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																								Aff3								
RES1	U	RES0						MT	Aff2								Aff1								Aff0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

Affinity level 3. See the description of VMPIDR_EL2.Aff0 for more information.

Aff3 is not supported in AArch32 state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [31]

Reserved, RES1.

U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system.

U	Meaning
0b0	Processor is part of a multiprocessor system.
0b1	Processor is part of a uniprocessor system.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [29:25]

Reserved, RES0.

MT, bit [24]

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of VMPIDR_EL2.Aff0 for more information about affinity levels.

MT	Meaning
0b0	Performance of PEs at the lowest affinity level is largely independent.
0b1	Performance of PEs at the lowest affinity level is very interdependent.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Aff2, bits [23:16]

Affinity level 2. See the description of VMPIDR_EL2.Aff0 for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Aff1, bits [15:8]

Affinity level 1. See the description of VMPIDR_EL2.Aff0 for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Aff0, bits [7:0]

Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior.

The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or [MPIDR_EL1](#).{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VMPIDR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VMPIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x050];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VMPIDR_EL2;
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        X[t, 64] = MPIDR_EL1;
    else
        X[t, 64] = VMPIDR_EL2;

```

MSR VMPIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x050] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VMPIDR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        return;
    else
        VMPIDR_EL2 = X[t, 64];

```

MRS <Xt>, MPIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b101


```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.MPIDR_EL1 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() then
            X[t, 64] = VMPIDR_EL2;
        else
            X[t, 64] = MPIDR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = MPIDR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = MPIDR_EL1;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VNCR_EL2, Virtual Nested Control Register

The VNCR_EL2 characteristics are:

Purpose

When FEAT_NV2 is implemented, holds the base address that is used to define the memory location that is accessed by transformed reads and writes of System registers.

Configuration

This register is present only when FEAT_NV2 is implemented. Otherwise, direct accesses to VNCR_EL2 are UNDEFINED.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VNCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RESS							BADDR																									
BADDR												RES0																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RESS, bits [63:57:53]

Reserved, Sign extended. If the bits marked as RESS do not all have the same value, then there is a CONSTRAINED UNPREDICTABLE choice between:

- Generating an EL2 translation regime Translation abort on use of the VNCR_EL2 register. If FEAT_D128 is implemented:
- If the virtual address space for EL2 supports 56 bits, bits[63:57] of VNCR_EL2 are treated as the same value as bit[56] for all purposes other than reading back the register.
- If the virtual address space for EL2 supports 56 bits, bits[63:57] of VNCR_EL2 are treated as the same value as bit[56].
- If the virtual address space for EL2 supports 52 bits, bits[63:53] of VNCR_EL2 are treated as the same value as bit[52] for all purposes other than reading back the register.
- If the virtual address space for EL2 supports 52 bits, bits[63:53] of VNCR_EL2 are treated as the same value as bit[52].
- Bits[63:49] of VNCR_EL2 are treated as the same value as bit[48] for all purposes other than reading back the register.
- Bits[63:49] of VNCR_EL2 are treated as the same value as bit[48] for all purposes.
- If the virtual address space for EL2 supports more than 48 bits, bits[63:53] of VNCR_EL2 are treated as the same value as bit[52] for all purposes other than reading back the register.
- If the virtual address space for EL2 supports more than 48 bits, bits[63:53] of VNCR_EL2 are treated as the same value as bit[52].

Where the EL2 translation regime has upper and lower address ranges, bit[56:52] is used to select between those address ranges to determine if the number address of space bits supports supported more by than the 48 address space bits.

BADDR, bits [56:12]

Base Address. If the virtual address space for EL2 does not support more than 48 bits, then bits [56:49] are RESS. If the virtual address space for EL2 does not support more than 52 bits, then bits [56:53] are RESS

When a register read/write is transformed to be a Load or Store, the address of the load/store is to SignOffset(VNCR_EL2.BADDR:Offset<11:0>, 64).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:0]

Reserved, RES0.

Accessing VNCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VNCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x0B0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VNCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VNCR_EL2;

```

MSR VNCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x0B0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VNCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VNCR_EL2 = X[t, 64];

```

VNCR_EL2, Virtual Nested Control Register

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VPIDR_EL2, Virtualization Processor ID Register

The VPIDR_EL2 characteristics are:

Purpose

Holds the value of the Virtualization Processor ID. This is the value returned by EL1 reads of [MIDR_EL1](#).

Configuration

AArch64 System register VPIDR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [VPIDR\[31:0\]](#).

If EL2 is not implemented, reads of this register return the value of the [MIDR_EL1](#) and writes to the register are ignored.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VPIDR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
RES0																																	
Implementer								Variant				Architecture				PartNum														Revision			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Bits [63:32]

Reserved, RES0.

Implementer, bits [31:24]

The Implementer code. This field must hold an implementer code that has been assigned by Arm. Assigned codes include the following:

Implementer	Meaning
0x00	Reserved for software use.
0x41	Arm Limited.
0x42	Broadcom Corporation.
0x43	Cavium Inc.
0x44	Digital Equipment Corporation.
0x46	Fujitsu Ltd.
0x49	Infineon Technologies AG.
0x4D	Motorola or Freescale Semiconductor Inc.
0x4E	NVIDIA Corporation.
0x50	Applied Micro Circuits Corporation.
0x51	Qualcomm Inc.
0x56	Marvell International Ltd.
0x69	Intel Corporation.
0xC0	Ampere Computing.

Arm can assign codes that are not published in this manual. All values not assigned by Arm are reserved and must not be used.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Architecture, bits [19:16]

Architecture version. Defined values are:

Architecture	Meaning
0b0001	Armv4.
0b0010	Armv4T.
0b0011	Armv5 (obsolete).
0b0100	Armv5T.
0b0101	Armv5TE.
0b0110	Armv5TEJ.
0b0111	Armv6.
0b1111	Architectural features are individually identified in the ID * registers.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VPIDR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VPIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x088];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VPIDR_EL2;
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        X[t, 64] = MIDR_EL1;
    else
        X[t, 64] = VPIDR_EL2;

```

MSR VPIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x088] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VPIDR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        return;
    else
        VPIDR_EL2 = X[t, 64];

```

MRS <Xt>, MIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b000

```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() then
        X[t, 64] = VPIDR_EL2;
    else
        X[t, 64] = MIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
elseif PSTATE.EL == EL3 then
    X[t, 64] = MIDR_EL1;

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VSTCR_EL2, Virtualization Secure Translation Control Register

The VSTCR_EL2 characteristics are:

Purpose

The control register for stage 2 of the Secure EL1&0 translation regime.

Configuration

This register is present only when FEAT_SEL2 is implemented. Otherwise, direct accesses to VSTCR_EL2 are UNDEFINED.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VSTCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
RES0																																	SL2	RES0
RES1	SA	SW	RES0													TG0	RES0						SL0	T0SZ										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

Any of the bits in VSTCR_EL2 are permitted to be cached in a TLB.

Bits [63:34]

Reserved, RES0.

SL2, bit [33]

When **FEAT_D128** is implemented and **VTCR_EL2.D128 == 1**:

This field is IGNORED.

When **FEAT_LPA2** is implemented and (**FEAT_D128** is not implemented or **VTCR_EL2.D128 == 0**):

Starting level of the Secure stage 2 translation lookup controlled by VSTCR_EL2.

If **VTCR_EL2.DS == 1**, then VSTCR_EL2.SL2, in combination with VSTCR_EL2.SL0, gives encodings for the Secure stage 2 translation table walk initial lookup level.

If **VTCR_EL2.DS == 0**, then VSTCR_EL2.SL2 is RES0.

If the translation granule size is not 4KB, then this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [32]

Reserved, RES0.

Bit [31]

Reserved, RES1.

SA, bit [30]

Secure stage 2 translation output address space.

SA	Meaning
0b0	All stage 2 translations for the Secure IPA space access the Secure PA space.
0b1	All stage 2 translations for the Secure IPA space access the Non-secure PA space.

When the value of VSTCR_EL2.SW is 1, this bit behaves as 1 for all purposes other than reading back the value of the bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SW, bit [29]

Secure stage 2 translation address space.

SW	Meaning
0b0	All stage 2 translation table walks for the Secure IPA space are to the Secure PA space.
0b1	All stage 2 translation table walks for the Secure IPA space are to the Non-secure PA space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [28:16]

Reserved, RES0.

TG0, bits [15:14]

Secure stage 2 granule size for [VSTTBR_EL2](#).

TG0	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

Other values are reserved.

If FEAT_GTG is implemented, [ID_AA64MMFR0_EL1](#).{TGran4_2, TGran16_2, TGran64_2} indicate which granule sizes are supported for stage 2 translation.

If FEAT_GTG is not implemented, [ID_AA64MMFR0_EL1](#).{TGran4, TGran16, TGran64} indicate which granule sizes are supported.

If the value is programmed to either a reserved value, or a size that has not been implemented, then for all purposes other than read back from this register, the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [13:8]

Reserved, RES0.

SL0, bits [7:6]

When FEAT_TTST is implemented and (FEAT_D128 is not implemented or VTCR_EL2.D128 == 0):

Starting level of the Secure stage 2 translation lookup, controlled by VSTCR_EL2. The meaning of this field depends on the value of VSTCR_EL2.TG0.

SL0	Meaning
0b00	If VSTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> If FEAT_LPA2 is not implemented, start at level 2. If FEAT_LPA2 is implemented and VSTCR_EL2.SL2 is 0b0, start at level 2. If FEAT_LPA2 is implemented and VSTCR_EL2.SL2 is 0b1, start at level -1. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 3.
0b01	If VSTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> If FEAT_LPA2 is not implemented, start at level 1. If FEAT_LPA2 is implemented and VSTCR_EL2.SL2 is 0b0, start at level 1. If FEAT_LPA2 is implemented, the combination of VSTCR_EL2.SL0 == 01 and VSTCR_EL2.SL2 == 1 is reserved. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 2.
0b10	If VSTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> If FEAT_LPA2 is not implemented, start at level 0. If FEAT_LPA2 is implemented and VSTCR_EL2.SL2 is 0b0, start at level 0. If FEAT_LPA2 is implemented, the combination of VSTCR_EL2.SL0 == 10 and VSTCR_EL2.SL2 == 1 is reserved. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 1.
0b11	If VSTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> If FEAT_LPA2 is not implemented, start at level 3. If FEAT_LPA2 is implemented and VSTCR_EL2.SL2 is 0b0, start at level 3. If FEAT_LPA2 is implemented, the combination of VSTCR_EL2.SL0 == 11 and VSTCR_EL2.SL2 == 1 is reserved. If VSTCR_EL2.TG0 is 0b10 (16KB granule) and FEAT_LPA2 is implemented, start at level 0.

If this field is programmed to a value that is not consistent with the programming of VSTCR_EL2.T0SZ, then a stage 2 level 0 Translation fault is generated.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Starting level of the Secure stage 2 translation lookup, controlled by VSTCR_EL2. The meaning of this field depends on the value of VSTCR_EL2.TG0.

SL0	Meaning
0b00	If VSTCR_EL2.TG0 is 0b00 (4KB granule), start at level 2. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 3.
0b01	If VSTCR_EL2.TG0 is 0b00 (4KB granule), start at level 1. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 2.
0b10	If VSTCR_EL2.TG0 is 0b00 (4KB granule), start at level 0. If VSTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 1.

All other values are reserved. If this field is programmed to a reserved value, or to a value that is not consistent with the programming of VSTCR_EL2.T0SZ, then a stage 2 level 0 Translation fault is generated.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [VSTTBR_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for this field depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

If this field is programmed to a value that is not consistent with the programming of SL0, then a stage 2 level 0 Translation fault is generated.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VSTCR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VSTCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x048];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        X[t, 64] = VSTCR_EL2;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        X[t, 64] = VSTCR_EL2;

```

MSR VSTCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x048] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        VSTCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        VSTCR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VSTTBR_EL2, Virtualization Secure Translation Table Base Register

The VSTTBR_EL2 characteristics are:

Purpose

The base register for stage 2 of the Secure EL1&0 translation regime. Holds the base address of the translation table for the initial lookup for stage 2 of an address translation in the Secure EL1&0 translation regime, and other information for this translation stage.

Configuration

This register is present only when FEAT_SEL2 is implemented. Otherwise, direct accesses to VSTTBR_EL2 are UNDEFINED.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VSTTBR_EL2 is a 64-bit register.

Field descriptions

When FEAT_D128 is implemented and VTCR_EL2.D128 == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0								BADDR																							
BADDR																								RES0CnP		SKL		CnP			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:56]

Reserved, RES0.

BADDR, bits [55:5]

- Bits A[55:x] of the stage 2 translation table base address bits are in register bits[55:x].
- Bits A[(x-1):0] of the stage 2 translation table base address are zero.

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on LOG2(StartTableSize), as described in VMSAv9-128. The smallest permitted value of x is 5.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level. Skip Level determines the number of levels to be skipped from the regular start level of the Secure Stage 2 translation table walk.

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

Common not Private, for stage 2 of the Secure EL1&0 translation regime. In an implementation that includes FEAT_TTCNP, indicates whether each entry that is pointed to by VSTTBR_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by VSTTBR_EL2 are permitted to differ from the entries for VSTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
0b1	The translation table entries pointed to by VSTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

This bit is permitted to be cached in a TLB.

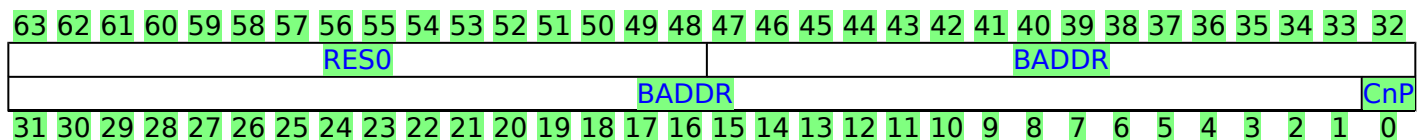
Note

If the value of VSTTBR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VSTTBR_EL2s do not point to the same translation table entries when using the current VMID, then the results of translations using VSTTBR_EL2 are CONstrained UNPREDICTABLE, see 'CONstrained UNPREDICTABLE behaviors due to caching of control or data values'.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is not implemented or VTCR_EL2.D128 == 0:**Bits [63:48]**

Reserved, RES0.

BADDR, bits [47:1]

Translation table base address, A[47:x] or A[51:x].

Note

A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

If the value of [VTCR_EL2](#).PS is 0b110, then:

- Register bits[47:z] hold bits[47:z] of the stage [21](#) translation table base address, where z is determined as follows:
 - If $x \geq 6$ then $z=x$.
 - Otherwise, $z=6$.
- Register bits[5:2] hold bits[51:48] of the stage [21](#) translation table base address.
- When $z > x$ register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When $x > 6$ register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage [21](#) translation table base address are zero.

Note

When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation does not support a 52-bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the Effective value of [VTCR_EL2](#).PS is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated. When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation supports a 56 bit PA size, bits [55:52] of the stage 2 translation table base address are zero.

If the Effective value of [VTCR_EL2](#).PS is not 0b110, then:

- Register bits[47:x] hold bits[47:x] of the stage [21](#) translation table base address.
- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:48] of the translation table base addresses used in this stage of translation are 0b0000.

If any VSTTBR_EL2[47:1] bit that is defined as RES0 has the value 1 when a translation table walk is performed using VSTTBR_EL2, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [VSTCR_EL2](#).T0SZ, the stage of translation, and the translation granule size.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]

Common not Private, for stage 2 of the Secure EL1&0 translation regime. In an implementation that includes FEAT_TTCNP, indicates whether each entry that is pointed to by VSTTBR_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by VSTTBR_EL2 are permitted to differ from the entries for VSTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
0b1	The translation table entries pointed to by VSTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VSTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

This bit is permitted to be cached in a TLB.

Note

If the value of VSTTBR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VSTTBR_EL2s do not point to the same translation table entries when using the current VMID, then the results of translations using VSTTBR_EL2 are **CONSTRAINED UNPREDICTABLE**, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VSTTBR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VSTTBR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x030];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        X[t, 64] = VSTTBR_EL2;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        X[t, 64] = VSTTBR_EL2;

```

MSR VSTTBR_EL2, <Xt>

op0	op1	CRn	CRm	op2
-----	-----	-----	-----	-----

0b11	0b100	0b0010	0b0110	0b000
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x030] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        VSTTBR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        VSTTBR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VTCR_EL2, Virtualization Translation Control Register

The VTCR_EL2 characteristics are:

Purpose

The control register for stage 2 of the EL1&0 translation regime.

Configuration

AArch64 System register VTCR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [VTCR\[31:0\]](#).

If EL2 is not implemented, this register is RES0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

VTCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37
RES0																			HAFTSL2	RES0DS	TL0	RES0D1	28	S2PO		
RES1	NSA	NSW	HWU62	HWU61	HWU60	HWU59	RES0	HD	HA	RES0VS	PS	TG0	SH0	ORGNO	IRGNO	SLO										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5

Unless stated otherwise, anyAny of the bits in VTCR_EL2 are permitted to be cached in a TLB.

Bits [63:4534]

Reserved, RES0.

HAFTSL2, bit [4433]

When **FEAT_HAFTFEAT_LPA2** is implemented:

Hardware managed Access Flag for Tables. Enables the Hardware managed Access Flag for Tables.

HAFT	Meaning
0b0	Hardware managed Access Flag for Tables is disabled.
0b1	Hardware managed Access Flag for Tables is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [43:42]

Reserved, RES0.

TL0, bit [41]**When FEAT_THE is implemented:**

Control bit to check for presence of MMU TopLevel0 permission attribute.

TL0	Meaning
0b0	This bit does not have any effect on Stage 2 translations.
0b1	Enables MMU TopLevel0 permission attribute check for TTBR0_EL1 and TTBR1_EL1 translations.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [40:39]

Reserved, RES0.

D128, bit [38]**When FEAT_D128 is implemented:**

Enable 128-bit Page Table Descriptors. Enables VMSAv9-128 translation system for the Stage 2 Translation Process.

D128	Meaning
0b0	Translation system follows VMSA-64 translation process.
0b1	Translation system follows VMSAv9-128 translation process.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S2POE, bit [37]**When FEAT_S2POE is implemented:**

Enable Permission Overlay. Enables permission overlay in Stage 2 Permission model.

S2POE	Meaning
0b0	Overlay disabled.
0b1	Overaly enabled.

This bit is not permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S2PIE, bit [36]**When FEAT_S2PIE is implemented:**

Select Permission Model. Enables usage of permission indirection in Stage 2 Permission model.

S2PIE	Meaning
0b0	Direct permission model.
0b1	Indirect permission model.

This field is RES1 when VTCR_EL2.D128 is set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TL1, bit [35]**When FEAT_THE is implemented:**

Control bit to check for presence of MMU TopLevel1 permission attribute.

TL1	Meaning
0b0	This bit does not have any effect on Stage 2 translations.
0b1	Enables MMU TopLevel1 permission attribute check for TTBR0_EL1 and TTBR1_EL1 translations.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

AssuredOnly, bit [34]**When FEAT_THE is implemented:**

AssuredOnly attribute enable. Indicates use of bit[58] of the stage 2 translation table block or page descriptor.

AssuredOnly	Meaning
0b0	Bit[58] of each stage 2 translation block or page descriptor do not indicate AssuredOnly attribute.
0b1	Bit[58] of each stage 2 translation block or page descriptor indicate AssuredOnly attribute.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SL2, bit [33]

When FEAT_LPA2 is implemented and (FEAT_D128 is not implemented or VTCR_EL2.D128 == 0):

Starting level of the stage 2 translation lookup controlled by VTCR_EL2.

If VTCR_EL2.DS == 1, then VTCR_EL2.SL2, in combination with VTCR_EL2.SL0, gives encodings for the stage 2 translation table walk initial lookup level.

If VTCR_EL2.DS == 0, then VTCR_EL2.SL2 is RES0.

If the translation granule size is not 4KB, then this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

When FEAT_D128 is implemented and VTCR_EL2.D128 == 1:

This field is IGNORED.

Otherwise:

Reserved, RES0.

DS, bit [32]

When FEAT_LPA2 is implemented:

This field affects whether a 52-bit output address addressing can be used when described by 4KB and the 16KB translation granules in stage 2 of the EL1&0 or 16KB translation granules regime.

DS	Meaning
0b0	<p>Bits[49:48] of translation descriptors are RES0.</p> <p>Bits[9:8] in Block and Page descriptors encode shareability information in the SH[1:0] field. Bits[9:8] in Table descriptors are ignored by hardware.</p> <p>The minimum value of VTCR_EL2.T0SZ is 16. Any memory access using a smaller value generates a stage 2 level 0 translation table fault.</p> <p>The minimum value of VSTCR_EL2.T0SZ is 16. Any memory access using a smaller value generates a stage 2 level 0 translation table fault.</p> <p>Output address[51:48] is 0000.</p>
0b1	<p>Bits[49:48] of translation descriptors hold output address[49:48].</p> <p>Bits[9:8] in translation descriptors hold output address[51:50].</p> <p>The shareability information of Block and Page descriptors for cacheable locations is determined by VTCR_EL2.SH0.</p> <p>The minimum value of VTCR_EL2.T0SZ is 12. Any memory access using a smaller value generates a stage 2 level 0 translation table fault.</p> <p>The minimum value of VSTCR_EL2.T0SZ is 12. Any memory access using a smaller value generates a stage 2 level 0 translation table fault.</p>
<p>Note</p> <p>As FEAT_LPA must be implemented if VTCR_EL2.DS == 1, the minimum values of VTCR_EL2.T0SZ and VSTCR_EL2.T0SZ are 12, as determined by that extension.</p> <p>For the TLBI range instructions affecting IPA, the format of the argument is changed so that bits[36:0] hold BaseADDR[52:16]. For the 4KB translation granule, bits[15:12] of BaseADDR are treated as 0000. For the 16KB translation granule, bits[15:14] of BaseADDR are treated as 00.</p>	
<p>Note</p> <p>This forces alignment of the ranges used by the TLBI range instructions.</p>	

This field is RES0 for a 64KB translation granule.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [31]

Reserved, RES1.

NSA, bit [30]

When FEAT_SEL2 is implemented:

Non-secure stage 2 translation output address space for the Secure EL1&0 translation regime.

NSA	Meaning
0b0	All stage 2 translations for the Non-secure IPA space of the Secure EL1&0 translation regime access the Secure PA space.
0b1	All stage 2 translations for the Non-secure IPA space of the Secure EL1&0 translation regime access the Non-secure PA space.

This bit behaves as 1 for all purposes other than reading back the value of the bit when one of the following is true:

- The value of VTCR_EL2.NSW is 1.
- The value of [VSTCR_EL2.SA](#) is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSW, bit [29]

When FEAT_SEL2 is implemented:

Non-secure stage 2 translation table address space for the Secure EL1&0 translation regime.

NSW	Meaning
0b0	All stage 2 translation table walks for the Non-secure IPA space of the Secure EL1&0 translation regime are to the Secure PA space.
0b1	All stage 2 translation table walks for the Non-secure IPA space of the Secure EL1&0 translation regime are to the Non-secure PA space.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU62, bit [28]

When FEAT_HPDS2 is implemented:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 2 translation table Block or Page entry.

HWU62	Meaning
0b0	Bit[62] of each stage 2 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[62] of each stage 2 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU61, bit [27]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 2 translation table Block or Page entry.

HWU61	Meaning
0b0	Bit[61] of each stage 2 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[61] of each stage 2 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU60, bit [26]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 2 translation table Block or Page entry.

HWU60	Meaning
0b0	Bit[60] of each stage 2 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[60] of each stage 2 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HWU59, bit [25]**When FEAT_HPDS2 is implemented:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 2 translation table Block or Page entry.

HWU59	Meaning
0b0	Bit[59] of each stage 2 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
0b1	Bit[59] of each stage 2 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [24:23]

Reserved, RES0.

HD, bit [22]**When FEAT_HAFDBS is implemented:**

Hardware management of dirty state in stage 2 translations when EL2 is enabled in the current Security state.

HD	Meaning
0b0	Stage 2 hardware management of dirty state disabled.
0b1	Stage 2 hardware management of dirty state enabled, only if the VTCR_EL2.HA bit is also set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

HA, bit [21]**When FEAT_HAFDBS is implemented:**

Hardware Access flag update in stage 2 translations when EL2 is enabled in the current Security state.

HA	Meaning
0b0	Stage 2 Access flag update disabled.
0b1	Stage 2 Access flag update enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [20]

Reserved, RES0.

VS, bit [19]**When FEAT_VMID16 is implemented:**

VMID Size.

VS	Meaning
0b0	8-bit VMID. The upper 8 bits of VTTBR_EL2 are ignored by the hardware, and treated as if they are all zeros, for every purpose except when reading back the register.
0b1	16-bit VMID. The upper 8 bits of VTTBR_EL2 are used for allocation and matching in the TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PS, bits [18:16]

Physical address Size for the second stage of translation.

PS	Meaning	Applies when
0b000	32 bits, 4GB.	
0b001	36 bits, 64GB.	
0b010	40 bits, 1TB.	
0b011	42 bits, 4TB.	
0b100	44 bits, 16TB.	
0b101	48 bits, 256TB.	
0b110	52 bits, 4PB.	
0b111	56 bits, 64PB.	When FEAT_D128 is implemented

All other values are reserved.

The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

If the translation granule is not 64KB and FEAT_LPA2 is not implemented, the value 0b110 is treated as reserved.

It is IMPLEMENTATION DEFINED whether an implementation that does not implement FEAT_LPA supports setting the value of 0b110 for the 64KB translation granule size or whether setting this value behaves as the 0b101 encoding.

In an implementation that supports 52-bit PAs, if the value of this field is not 0b110 or a value treated as 0b110, then bits[51:48] of every translation table base address for the stage of translation controlled by VTCR_EL2 are 0b0000.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

TG0, bits [15:14]

Granule size for the [VTTBR_EL2](#).

TG0	Meaning
0b00	4KB.
0b01	64KB.
0b10	16KB.

Other values are reserved.

If FEAT_GTG is implemented, [ID_AA64MMFR0_EL1](#).{TGran4_2, TGran16_2, TGran64_2} indicate which granule sizes are supported for stage 2 translation.

If FEAT_GTG is not implemented, [ID_AA64MMFR0_EL1](#).{TGran4, TGran16, TGran64} indicate which granule sizes are supported.

If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [VTTBR_EL2](#) or [VSTTBR_EL2](#).

SH0	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [VTTBR_EL2](#) or [VSTTBR_EL2](#).

ORGN0	Meaning
0b00	Normal memory, Outer Non-cacheable.
0b01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [VTTBR_EL2](#) or [VSTTBR_EL2](#).

IRGN0	Meaning
0b00	Normal memory, Inner Non-cacheable.
0b01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.
0b10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.
0b11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

SLO, bits [7:6]

When FEAT_TTST is implemented and (FEAT_D128 is not implemented or VTCR_EL2.D128 == 0):

Starting level of the stage 2 translation lookup, controlled by VTCR_EL2. The meaning of this field depends on the value of VTCR_EL2.TG0.

SL0	Meaning
0b00	If VTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> • If FEAT_LPA2 is not implemented, start at level 2. • If FEAT_LPA2 is implemented and VTCR_EL2.SL2 is 0b0, start at level 2. • If FEAT_LPA2 is implemented and VTCR_EL2.SL2 is 0b1, start at level -1. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 3.
0b01	If VTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> • If FEAT_LPA2 is not implemented, start at level 1. • If FEAT_LPA2 is implemented and VTCR_EL2.SL2 is 0b0, start at level 1. • If FEAT_LPA2 is implemented, the combination of VTCR_EL2.SL0 == 01 and VTCR_EL2.SL2 == 1 is reserved. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 2.
0b10	If VTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> • If FEAT_LPA2 is not implemented, start at level 0. • If FEAT_LPA2 is implemented and VTCR_EL2.SL2 is 0b0, start at level 0. • If FEAT_LPA2 is implemented, the combination of VTCR_EL2.SL0 == 10 and VTCR_EL2.SL2 == 1 is reserved. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 1.
0b11	If VTCR_EL2.TG0 is 0b00 (4KB granule): <ul style="list-style-type: none"> • If FEAT_LPA2 is not implemented, start at level 3. • If FEAT_LPA2 is implemented and VTCR_EL2.SL2 is 0b0, start at level 3. • If FEAT_LPA2 is implemented, the combination of VTCR_EL2.SL0 == 11 and VTCR_EL2.SL2 == 1 is reserved. If VTCR_EL2.TG0 is 0b10 (16KB granule) and FEAT_LPA2 is implemented, start at level 0.

If this field is programmed to a value that is not consistent with the programming of VTCR_EL2.T0SZ, then a stage 2 level 0 Translation fault is generated.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Starting level of the stage 2 translation lookup, controlled by VTCR_EL2. The meaning of this field depends on the value of VTCR_EL2.TG0.

SL0	Meaning
0b00	If VTCR_EL2.TG0 is 0b00 (4KB granule), start at level 2. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 3.
0b01	If VTCR_EL2.TG0 is 0b00 (4KB granule), start at level 1. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 2.
0b10	If VTCR_EL2.TG0 is 0b00 (4KB granule), start at level 0. If VTCR_EL2.TG0 is 0b10 (16KB granule) or 0b01 (64KB granule), start at level 1.

All other values are reserved. If this field is programmed to a reserved value, or to a value that is not consistent with the programming of VTCR_EL2.T0SZ, then a stage 2 level 0 Translation fault is generated.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [VTTBR_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in 'The AArch64 Virtual Memory System Architecture'.

If this field is programmed to a value that is not consistent with the programming of SL0, then a stage 2 level 0 Translation fault is generated.

Note

For the 4KB translation granule, if FEAT_LPA2 is implemented and this field is less than 16, the translation table walk begins with a level -1 initial lookup.

For the 16KB translation granule, if FEAT_LPA2 is implemented and this field is less than 17, the translation table walk begins with a level 0 initial lookup.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VTCR_EL2

Unless stated otherwise, any of the bits in VTCR_EL2 are permitted to be cached in a TLB.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VTCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x040];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VTCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VTCR_EL2;

```

MSR VTCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x040] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VTCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VTCR_EL2 = X[t, 64];

```

3005/0907/2022 1517:5807: 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Address bit x is the minimum address bit required to align the translation table to the size of the table. x is calculated based on $\text{LOG2}(\text{StartTableSize})$, as described in VMSAv9-128. The smallest permitted value of x is 5.

The BADDR field is split as follows:

- BADDR[50:43] is VTTBR_EL2[87:80].
- BADDR[42:0] is VTTBR_EL2[47:5].

The reset behavior of this field is:

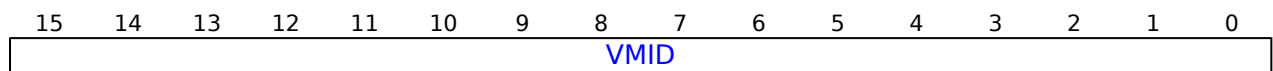
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [79:64]

Reserved, RES0.

VMID, bits [63:48]

VMID encoding when FEAT_VMID16 is implemented and VTCR_EL2.VS == 1



VMID, bits [15:0]

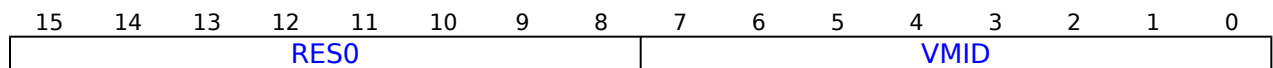
The VMID for the translation table.

If the implementation has an 8-bit VMID, bits [15:8] of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VMID encoding when FEAT_VMID16 is not implemented or VTCR_EL2.VS == 0



Bits [15:8]

Reserved, RES0.

VMID, bits [7:0]

The VMID for the translation table.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- VTCR_EL2.VS is 0.
- FEAT_VMID16 is not implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

SKL, bits [2:1]

Skip Level. Skip Level determines the number of levels to be skipped from the regular start level of the Non-Secure Stage 2 translation table walk.

SKL	Meaning
0b00	Skip 0 level from the regular start level.
0b01	Skip 1 level from the regular start level.
0b10	Skip 2 levels from the regular start level.
0b11	Skip 3 levels from the regular start level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by VTTBR_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VTTBR_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by VTTBR_EL2 are permitted to differ from the entries for VTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
0b1	The translation table entries pointed to by VTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

This bit is permitted to be cached in a TLB.

Note

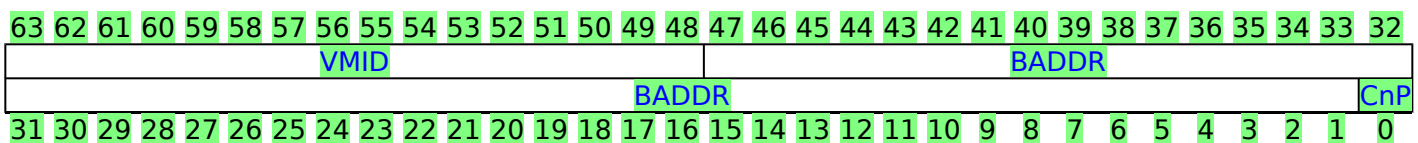
If the value of VTTBR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VTTBR_EL2s do not point to the same translation table entries when using the current VMID then the results of translations using VTTBR_EL2 are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values'.

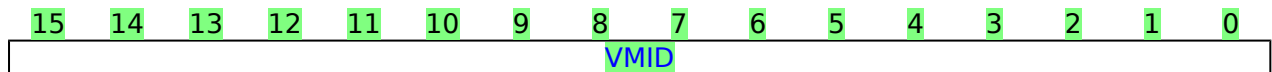
The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

When FEAT_D128 is not implemented or VTCR_EL2.D128 == 0:

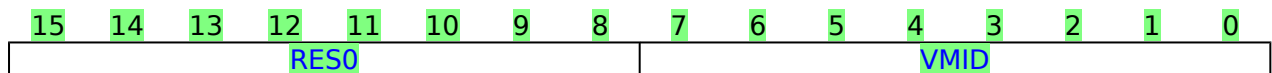
VMID, bits [63:48]**VMID encoding when FEAT_VMID16 is implemented and VTCR_EL2.VS == 1****VMID, bits [15:0]**

The VMID for the translation table.

If the implementation has an 8-bit VMID, bits [15:8] of this field are RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

VMID encoding when FEAT_VMID16 is not implemented or VTCR_EL2.VS == 0**Bits [15:8]**

Reserved, RES0.

VMID, bits [7:0]

The VMID for the translation table.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- VTCR_EL2.VS is 0.
- FEAT_VMID16 is not implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

BADDR, bits [47:1]

Translation table base address, A[47:x] or A[51:x], bits[47:1].

Note

A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes FEAT_LPA, if the value of VTCR_EL2.PS is 0b110, then:

- Register bits[47:z] hold bits[47:z] of the stage 2¹ translation table base address, where z is determined as follows:
 - If $x \geq 6$ then $z=x$.
 - Otherwise, $z=6$.
- Register bits[5:2] hold bits[51:48] of the stage 2¹ translation table base address.
- When $z > x$ register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.

- When $x > 6$ register bits[($x-1$):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 2¹ translation table base address are zero.
- In an implementation that includes FEAT_TTCNP, bit[0] of the stage 2¹ translation table base address is zero.

If the Effective value of [VTCR_EL2](#).PS is not 0b110 then:

- Register bits[47: x] hold bits[47: x] of the stage 1 translation table base address.
- Register bits[($x-1$):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:48] of the translation table base addresses used in this stage of translation are 0b0000.

If any VTTBR_EL2[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using VTTBR_EL2, then the translation table base address might be misaligned, with effects that are CONstrained UNPREDICTABLE, and must be one of the following:

- Bits[$x-1$:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [VTCR_EL2](#).T0SZ, the stage of translation, and the translation granule size.

Note

When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register when the Effective value of [VTCR_EL2](#).PS is 0b110 and the value of register bits[5:2] is nonzero, an Address size fault is generated. When the value of [ID_AA64MMFR0_EL1](#).PARange indicates that the implementation supports a 56 bit PA size, bits [55:52] of the stage 2 translation table base address are zero.

If the Effective value of [VTCR_EL2](#).PS is not 0b110 then:

- Register bits[47: x] hold bits[47: x] of the stage 2 translation table base address.
- Register bits[($x-1$):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:48] of the translation table base addresses used in this stage of translation are 0b0000.

If any VTTBR_EL2[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using VTTBR_EL2, then the translation table base address might be misaligned, with effects that are CONstrained UNPREDICTABLE, and must be one of the following:

- Bits[$x-1$:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [VTCR_EL2](#).T0SZ, the stage of translation, and the translation granule size.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CnP, bit [0]**When FEAT_TTCNP is implemented:**

Common not Private. This bit indicates whether each entry that is pointed to by VTTBR_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VTTBR_EL2.CnP is 1.

CnP	Meaning
0b0	The translation table entries pointed to by VTTBR_EL2 are permitted to differ from the entries for VTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
0b1	The translation table entries pointed to by VTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

This bit is permitted to be cached in a TLB.

Note

If the value of VTTBR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VTTBR_EL2s do not point to the same translation table entries when using the current VMID then the results of translations using VTTBR_EL2 are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing VTTBR_EL2

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, VTTBR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x020];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VTTBR_EL2<63:0>;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VTTBR_EL2<63:0>;

```

MSR VTTBR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x020] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VTTBR_EL2<63:0> = X[t, 64];
elsif PSTATE.EL == EL3 then
    VTTBR_EL2<63:0> = X[t, 64];

```

When FEAT_D128 is implemented

MRRS <Xt+1>, <Xt>, VTTBR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        (X[t + 1, 64], X[t, 64]) = (NVMem[0x028], NVMem[0x020]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
    else
        (X[t + 1, 64], X[t, 64]) = (VTTBR_EL2<127:64>, VTTBR_EL2<63:0>);
elsif PSTATE.EL == EL3 then
    (X[t + 1, 64], X[t, 64]) = (VTTBR_EL2<127:64>, VTTBR_EL2<63:0>);

```

When FEAT_D128 is implemented

MSRR VTTBR_EL2, <Xt+1>, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        (NVMem[0x028], NVMem[0x020]) = (X[t + 1, 64], X[t, 64]);
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.D128En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.D128En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x14);
        else
            (VTTBR_EL2<127:64>, VTTBR_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);
elsif PSTATE.EL == EL3 then
    (VTTBR_EL2<127:64>, VTTBR_EL2<63:0>) = (X[t + 1, 64], X[t, 64]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ZCR_EL1, SVE Control Register (EL1)

The ZCR_EL1 characteristics are:

Purpose

This register controls aspects of SVE visible at Exception levels EL1 and EL0.

Configuration

This register is present only when FEAT_SVE is implemented. Otherwise, direct accesses to ZCR_EL1 are UNDEFINED.

This register has no effect when FEAT_SME is implemented and the PE is in Streaming SVE mode.

When [HCR_EL2](#).{E2H, TGE} == {1, 1} and EL2 is enabled in the current Security state, this register has no effect on execution at EL0.

Attributes

ZCR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
RES0																RAZ/WI						LEN									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Non-streaming SVE vector length at EL1 of (LEN+1)*128 bits. This field also defines the Effective Non-streaming SVE vector length at EL0 when EL2 is not implemented, or EL2 is not enabled in the current Security state, or HCR_EL2.{E2H,TGE} is not {1,1}.

The Non-streaming SVE vector length can be any power multiple of two128 bits, from 128 bits to 2048 bits inclusive. An implementation can support a subset of the architecturally permitted lengths. An implementation is required to support all lengths that are powers of two, from 128 bits up to its maximum implemented Non-streaming SVE vector length.

When FEAT_SME is not implemented, or the PE is not in Streaming SVE mode, the Effective SVE vector length (VL) is equal to the Effective Non-streaming SVE vector length.

When FEAT_SME is implemented and the PE is in Streaming SVE mode, VL is equal to the Effective Streaming SVE vector length. See [SMCR_EL1](#).

For all purposes other than returning the result of a direct read of ZCR_EL1, the PE selects the Effective Non-streaming SVE vector length by performing checks in the following order:

1. If EL2 is implemented and enabled in the current Security state, and the requested length is greater than the Effective length at EL2, then the Effective length at EL2 is used.
2. If EL3 is implemented and the requested length is greater than the Effective length at EL3, then the Effective length at EL3 is used.
3. Otherwise, the Effective length is the highest supported Non-streaming SVE vector length that is less than or equal to the requested length.

An indirect read of ZCR_EL1.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ZCR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic ZCR_EL1 or ZCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ZCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            X[t, 64] = NVMem[0x1E0];
        else
            X[t, 64] = ZCR_EL1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
            UNDEFINED;
        elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x19);
        elsif HCR_EL2.E2H == '1' then
            X[t, 64] = ZCR_EL2;
        else
            X[t, 64] = ZCR_EL1;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.EZ == '0' then
            AArch64.SystemAccessTrap(EL3, 0x19);
        else
            X[t, 64] = ZCR_EL1;

```

MSR ZCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x1E0] = X[t, 64];
        else
            ZCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
            UNDEFINED;
        elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x19);
        elsif HCR_EL2.E2H == '1' then
            ZCR_EL2 = X[t, 64];
        else
            ZCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.EZ == '0' then
            AArch64.SystemAccessTrap(EL3, 0x19);
        else
            ZCR_EL1 = X[t, 64];

```

MRS <Xt>, ZCR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x1E0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
            UNDEFINED;
        elsif CPTR_EL2.ZEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x19);
        else
            X[t, 64] = ZCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CPTR_EL3.EZ == '0' then
            AArch64.SystemAccessTrap(EL3, 0x19);
        else
            X[t, 64] = ZCR_EL1;
    else
        UNDEFINED;

```

MSR ZCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x1E0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
            UNDEFINED;
        elsif CPTR_EL2.ZEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x19);
        else
            ZCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CPTR_EL3.EZ == '0' then
            AArch64.SystemAccessTrap(EL3, 0x19);
        else
            ZCR_EL1 = X[t, 64];
    else
        UNDEFINED;

```

3005/0907/2022 1517:5809: 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ZCR_EL2, SVE Control Register (EL2)

The ZCR_EL2 characteristics are:

Purpose

This register controls aspects of SVE visible at Exception levels EL2, EL1, and EL0.

Configuration

This register is present only when FEAT_SVE is implemented. Otherwise, direct accesses to ZCR_EL2 are UNDEFINED.

This register has no effect when EL2 is not enabled in the current Security state, or when FEAT_SME is implemented and the PE is in Streaming SVE mode.

Attributes

ZCR_EL2 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																								RAZ/WI				LEN			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Non-streaming SVE vector length at EL2 of (LEN+1)*128 bits. This field also defines the Effective Non-streaming SVE vector length at EL0 when EL2 is implemented and enabled in the current Security state, and HCR_EL2.{E2H,TGE} is {1,1}.

The Non-streaming SVE vector length can be any power multiple of two 128 bits, from 128 bits to 2048 bits inclusive. An implementation can support a subset of the architecturally permitted lengths. An implementation is required to support all lengths that are powers of two, from 128 bits up to its maximum implemented Non-streaming SVE vector length.

When FEAT_SME is not implemented, or the PE is not in Streaming SVE mode, the Effective SVE vector length (VL) is equal to the Effective Non-streaming SVE vector length.

When FEAT_SME is implemented and the PE is in Streaming SVE mode, VL is equal to the Effective Streaming SVE vector length. See [SMCR_EL2](#).

For all purposes other than returning the result of a direct read of ZCR_EL2, the PE selects the Effective Non-streaming SVE vector length by performing checks in the following order:

1. If EL3 is implemented and the requested length is greater than the Effective length at EL3, then the Effective length at EL3 is used.

2. Otherwise, the Effective length is the highest supported Non-streaming SVE vector length that is less than or equal to the requested length.

An indirect read of ZCR_EL2.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ZCR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic ZCR_EL2 or ZCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ZCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL2;

```

MSR ZCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL2 = X[t, 64];

```

MRS <Xt>, ZCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b000


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x1E0];
    else
        X[t, 64] = ZCR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = ZCR_EL2;
    else
        X[t, 64] = ZCR_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL1;

```

MSR ZCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elsif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x1E0] = X[t, 64];
        else
            ZCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && CPTR_EL3.EZ == '0' then
            UNDEFINED;
        elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0' then
            AArch64.SystemAccessTrap(EL2, 0x19);
        elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x19);
            elsif HCR_EL2.E2H == '1' then
                ZCR_EL2 = X[t, 64];
            else
                ZCR_EL1 = X[t, 64];
        elsif PSTATE.EL == EL3 then
            if CPTR_EL3.EZ == '0' then
                AArch64.SystemAccessTrap(EL3, 0x19);
            else
                ZCR_EL1 = X[t, 64];

```

3005/0907/2022 1517:5809: 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ZCR_EL3, SVE Control Register (EL3)

The ZCR_EL3 characteristics are:

Purpose

This register controls aspects of SVE visible at all Exception levels.

Configuration

This register is present only when FEAT_SVE is implemented. Otherwise, direct accesses to ZCR_EL3 are UNDEFINED.

This register has no effect when FEAT_SME is implemented and the PE is in Streaming SVE mode.

Attributes

ZCR_EL3 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																								RAZ/WI							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Requests an Effective Non-streaming SVE vector length at EL3 of (LEN+1)*128 bits.

The Non-streaming SVE vector length can be any power multiple of two128 bits, from 128 bits to 2048 bits inclusive. An implementation can support a subset of the architecturally permitted lengths. An implementation is required to support all lengths that are powers of two, from 128 bits up to its maximum implemented Non-streaming SVE vector length.

When FEAT_SME is not implemented, or the PE is not in Streaming SVE mode, the Effective SVE vector length (VL) is equal to the Effective Non-streaming SVE vector length.

When FEAT_SME is implemented and the PE is in Streaming SVE mode, VL is equal to the Effective Streaming SVE vector length. See [SMCR_EL3](#).

For all purposes other than returning the result of a direct read of ZCR_EL3, the PE selects the highest supported Non-streaming SVE vector length that is less than or equal to the requested length.

An indirect read of ZCR_EL3.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ZCR_EL3

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, ZCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL3;
    end
end

```

MSR ZCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR EL3 = X[t, 64];
```

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AArch32 System Registers

ACTLR: Auxiliary Control Register

ACTLR2: Auxiliary Control Register 2

ADFSR: Auxiliary Data Fault Status Register

AIDR: Auxiliary ID Register

AIFSR: Auxiliary Instruction Fault Status Register

AMAIR0: Auxiliary Memory Attribute Indirection Register 0

AMAIR1: Auxiliary Memory Attribute Indirection Register 1

AMCFGR: Activity Monitors Configuration Register

AMCGCR: Activity Monitors Counter Group Configuration Register

[AMCNTENCLR0](#): Activity Monitors Count Enable Clear Register 0

[AMCNTENCLR1](#): Activity Monitors Count Enable Clear Register 1

[AMCNTENSET0](#): Activity Monitors Count Enable Set Register 0

[AMCNTENSET1](#): Activity Monitors Count Enable Set Register 1

AMCR: Activity Monitors Control Register

[AMEVCNTR0<n>](#): Activity Monitors Event Counter Registers 0

[AMEVCNTR1<n>](#): Activity Monitors Event Counter Registers 1

AMEVTYPER0<n>: Activity Monitors Event Type Registers 0

[AMEVTYPER1<n>](#): Activity Monitors Event Type Registers 1

AMUSERENR: Activity Monitors User Enable Register

APSR: Application Program Status Register

CCSIDR: Current Cache Size ID Register

CCSIDR2: Current Cache Size ID Register 2

[CLIDR](#): Cache Level ID Register

CNTFRQ: Counter-timer Frequency register

CNTHCTL: Counter-timer Hyp Control register

CNTHPS_CTL: Counter-timer Secure Physical Timer Control Register (EL2)

CNTHPS_CVAL: Counter-timer Secure Physical Timer CompareValue Register (EL2)

CNTHPS_TVAL: Counter-timer Secure Physical Timer TimerValue Register (EL2)

CNTHP_CTL: Counter-timer Hyp Physical Timer Control register

CNTHP_CVAL: Counter-timer Hyp Physical CompareValue register

CNTHP_TVAL: Counter-timer Hyp Physical Timer TimerValue register

CNTHVS_CTL: Counter-timer Secure Virtual Timer Control Register (EL2)

CNTHVS_CVAL: Counter-timer Secure Virtual Timer CompareValue Register (EL2)

CNTHVS_TVAL: Counter-timer Secure Virtual Timer TimerValue Register (EL2)

CNTHV_CTL: Counter-timer Virtual Timer Control register (EL2)

CNTHV_CVAL: Counter-timer Virtual Timer CompareValue register (EL2)

CNTHV_TVAL: Counter-timer Virtual Timer TimerValue register (EL2)

CNTKCTL: Counter-timer Kernel Control register

CNTPCT: Counter-timer Physical Count register

CNTPCTSS: Counter-timer Self-Synchronized Physical Count register

CNTP_CTL: Counter-timer Physical Timer Control register

CNTP_CVAL: Counter-timer Physical Timer CompareValue register

CNTP_TVAL: Counter-timer Physical Timer TimerValue register

CNTVCT: Counter-timer Virtual Count register

CNTVCTSS: Counter-timer Self-Synchronized Virtual Count register

CNTVOFF: Counter-timer Virtual Offset register

CNTV_CTL: Counter-timer Virtual Timer Control register

CNTV_CVAL: Counter-timer Virtual Timer CompareValue register

CNTV_TVAL: Counter-timer Virtual Timer TimerValue register

CONTEXTIDR: Context ID Register

CPACR: Architectural Feature Access Control Register

CPSR: Current Program Status Register

CSSELR: Cache Size Selection Register

CTR: Cache Type Register

DACR: Domain Access Control Register

DBGAUTHSTATUS: Debug Authentication Status register

[DBGBCR<n>](#): Debug Breakpoint Control Registers

[DBGBVR<n>](#): Debug Breakpoint Value Registers

[DBGBXVR<n>](#): Debug Breakpoint Extended Value Registers

DBGCLAIMCLR: Debug CLAIM Tag Clear register

DBGCLAIMSET: Debug CLAIM Tag Set register

DBGDCCINT: DCC Interrupt Enable Register

[DBGDEVID](#): Debug Device ID register 0

[DBGDEVID1](#): Debug Device ID register 1

DBGDEVID2: Debug Device ID register 2

[DBGDIDR](#): Debug ID Register

DBGDRAR: Debug ROM Address Register

DBGDSAR: Debug Self Address Register

DBGDSCRext: Debug Status and Control Register, External View

DBGDSCRInt: Debug Status and Control Register, Internal View

DBGDTRRText: Debug OS Lock Data Transfer Register, Receive, External View

DBGDTRRXint: Debug Data Transfer Register, Receive

DBGDTRTXext: Debug OS Lock Data Transfer Register, Transmit

DBGDTRTXint: Debug Data Transfer Register, Transmit

DBGOSDLR: Debug OS Double Lock Register

DBGOSECCR: Debug OS Lock Exception Catch Control Register

DBGOSLAR: Debug OS Lock Access Register

DBGOSLSR: Debug OS Lock Status Register

DBGPRCR: Debug Power Control Register

DBGVCR: Debug Vector Catch Register

[DBGWCR<n>](#): Debug Watchpoint Control Registers

DBGWFAR: Debug Watchpoint Fault Address Register

DBGWVR<n>: Debug Watchpoint Value Registers

DFAR: Data Fault Address Register

DFSR: Data Fault Status Register

DISR: Deferred Interrupt Status Register

DLR: Debug Link Register

DSPSR: Debug Saved Program Status Register

[DSPSR2](#): Debug Saved Process State Register 2

ELR_hyp: Exception Link Register (Hyp mode)

ERRIDR: Error Record ID Register

ERRSELR: Error Record Select Register

ERXADDR: Selected Error Record Address Register

ERXADDR2: Selected Error Record Address Register 2

ERXCTLR: Selected Error Record Control Register

ERXCTLR2: Selected Error Record Control Register 2

ERXFR: Selected Error Record Feature Register

ERXFR2: Selected Error Record Feature Register 2

ERXMISC0: Selected Error Record Miscellaneous Register 0

ERXMISC1: Selected Error Record Miscellaneous Register 1

ERXMISC2: Selected Error Record Miscellaneous Register 2

ERXMISC3: Selected Error Record Miscellaneous Register 3

ERXMISC4: Selected Error Record Miscellaneous Register 4

ERXMISC5: Selected Error Record Miscellaneous Register 5

ERXMISC6: Selected Error Record Miscellaneous Register 6

ERXMISC7: Selected Error Record Miscellaneous Register 7

ERXSTATUS: Selected Error Record Primary Status Register

FCSEIDR: FCSE Process ID register

FPEXC: Floating-Point Exception Control register

FPSCR: Floating-Point Status and Control Register

FPSID: Floating-Point System ID register

HACR: Hyp Auxiliary Configuration Register

HACTLR: Hyp Auxiliary Control Register

HACTLR2: Hyp Auxiliary Control Register 2

HADFSR: Hyp Auxiliary Data Fault Status Register

HAIFSR: Hyp Auxiliary Instruction Fault Status Register

HAMAIRO: Hyp Auxiliary Memory Attribute Indirection Register 0

HAMAIR1: Hyp Auxiliary Memory Attribute Indirection Register 1

HCPTR: Hyp Architectural Feature Trap Register

HCR: Hyp Configuration Register

HCR2: Hyp Configuration Register 2

HDCR: Hyp Debug Control Register

HDFAR: Hyp Data Fault Address Register

HIFAR: Hyp Instruction Fault Address Register

HMAIRO: Hyp Memory Attribute Indirection Register 0

HMAIR1: Hyp Memory Attribute Indirection Register 1

HPFAR: Hyp IPA Fault Address Register

HRMR: Hyp Reset Management Register

HSCTLR: Hyp System Control Register

HSR: Hyp Syndrome Register

HSTR: Hyp System Trap Register

HTCR: Hyp Translation Control Register

HTPIDR: Hyp Software Thread ID Register

HTRFCR: Hyp Trace Filter Control Register

HTTBR: Hyp Translation Table Base Register

HVBAR: Hyp Vector Base Address Register

ICC_AP0R<n>: Interrupt Controller Active Priorities Group 0 Registers

[ICC_AP1R<n>](#): Interrupt Controller Active Priorities Group 1 Registers

ICC_ASGI1R: Interrupt Controller Alias Software Generated Interrupt Group 1 Register

ICC_BPR0: Interrupt Controller Binary Point Register 0

[ICC_BPR1](#): Interrupt Controller Binary Point Register 1

[ICC_CTLR](#): Interrupt Controller Control Register

ICC_DIR: Interrupt Controller Deactivate Interrupt Register

ICC_EOIR0: Interrupt Controller End Of Interrupt Register 0

ICC_EOIR1: Interrupt Controller End Of Interrupt Register 1

ICC_HPPIR0: Interrupt Controller Highest Priority Pending Interrupt Register 0

ICC_HPPIR1: Interrupt Controller Highest Priority Pending Interrupt Register 1

ICC_HSRE: Interrupt Controller Hyp System Register Enable register

ICC_IAR0: Interrupt Controller Interrupt Acknowledge Register 0

ICC_IAR1: Interrupt Controller Interrupt Acknowledge Register 1

ICC_IGRPEN0: Interrupt Controller Interrupt Group 0 Enable register

[ICC_IGRPEN1](#): Interrupt Controller Interrupt Group 1 Enable register

ICC_MCTLR: Interrupt Controller Monitor Control Register

ICC_MGRPEN1: Interrupt Controller Monitor Interrupt Group 1 Enable register

ICC_MSRE: Interrupt Controller Monitor System Register Enable register

ICC_PMR: Interrupt Controller Interrupt Priority Mask Register

ICC_RPR: Interrupt Controller Running Priority Register

ICC_SGI0R: Interrupt Controller Software Generated Interrupt Group 0 Register

ICC_SGI1R: Interrupt Controller Software Generated Interrupt Group 1 Register

[ICC_SRE](#): Interrupt Controller System Register Enable register

ICH_AP0R<n>: Interrupt Controller Hyp Active Priorities Group 0 Registers

ICH_AP1R<n>: Interrupt Controller Hyp Active Priorities Group 1 Registers

ICH_EISR: Interrupt Controller End of Interrupt Status Register

ICH_ELRSR: Interrupt Controller Empty List Register Status Register

ICH_HCR: Interrupt Controller Hyp Control Register

ICH_LR<n>: Interrupt Controller List Registers

ICH_LRC<n>: Interrupt Controller List Registers

ICH_MISR: Interrupt Controller Maintenance Interrupt State Register

ICH_VMCR: Interrupt Controller Virtual Machine Control Register

ICH_VTR: Interrupt Controller VGIC Type Register

ICV_AP0R<n>: Interrupt Controller Virtual Active Priorities Group 0 Registers

ICV_AP1R<n>: Interrupt Controller Virtual Active Priorities Group 1 Registers

ICV_BPR0: Interrupt Controller Virtual Binary Point Register 0

ICV_BPR1: Interrupt Controller Virtual Binary Point Register 1

[ICV_CTLR](#): Interrupt Controller Virtual Control Register

ICV_DIR: Interrupt Controller Deactivate Virtual Interrupt Register

ICV_EOIR0: Interrupt Controller Virtual End Of Interrupt Register 0

ICV_EOIR1: Interrupt Controller Virtual End Of Interrupt Register 1

ICV_HPPIR0: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

ICV_HPPIR1: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

ICV_IAR0: Interrupt Controller Virtual Interrupt Acknowledge Register 0

ICV_IAR1: Interrupt Controller Virtual Interrupt Acknowledge Register 1

ICV_IGRPEN0: Interrupt Controller Virtual Interrupt Group 0 Enable register

ICV_IGRPEN1: Interrupt Controller Virtual Interrupt Group 1 Enable register

ICV_PMR: Interrupt Controller Virtual Interrupt Priority Mask Register

ICV_RPR: Interrupt Controller Virtual Running Priority Register

ID_AFR0: Auxiliary Feature Register 0

[ID_DFR0](#): Debug Feature Register 0

ID_DFR1: Debug Feature Register 1

ID_ISAR0: Instruction Set Attribute Register 0

ID_ISAR1: Instruction Set Attribute Register 1

ID_ISAR2: Instruction Set Attribute Register 2

ID_ISAR3: Instruction Set Attribute Register 3

ID_ISAR4: Instruction Set Attribute Register 4

ID_ISAR5: Instruction Set Attribute Register 5

[ID_ISAR6](#): Instruction Set Attribute Register 6

ID_MMFR0: Memory Model Feature Register 0

ID_MMFR1: Memory Model Feature Register 1

ID_MMFR2: Memory Model Feature Register 2

ID_MMFR3: Memory Model Feature Register 3

ID_MMFR4: Memory Model Feature Register 4

ID_MMFR5: Memory Model Feature Register 5

[ID_PFR0](#): Processor Feature Register 0

ID_PFR1: Processor Feature Register 1

ID_PFR2: Processor Feature Register 2

IFAR: Instruction Fault Address Register

IFSR: Instruction Fault Status Register

ISR: Interrupt Status Register

JIDR: Jazelle ID Register

JMCR: Jazelle Main Configuration Register

JOSCR: Jazelle OS Control Register

MAIR0: Memory Attribute Indirection Register 0

MAIR1: Memory Attribute Indirection Register 1

MIDR: Main ID Register

MPIDR: Multiprocessor Affinity Register

MVBAR: Monitor Vector Base Address Register

MVFR0: Media and VFP Feature Register 0

MVFR1: Media and VFP Feature Register 1

MVFR2: Media and VFP Feature Register 2

NMRR: Normal Memory Remap Register

NSACR: Non-Secure Access Control Register

PAR: Physical Address Register

[PMCCFILTR](#): Performance Monitors Cycle Count Filter Register

[PMCCNTR](#): Performance Monitors Cycle Count Register

[PMCEID0](#): Performance Monitors Common Event Identification register 0

[PMCEID1](#): Performance Monitors Common Event Identification register 1

[PMCEID2](#): Performance Monitors Common Event Identification register 2

[PMCEID3](#): Performance Monitors Common Event Identification register 3

[PMCNTENCLR](#): Performance Monitors Count Enable Clear register

[PMCNTENSET](#): Performance Monitors Count Enable Set register

[PMCR](#): Performance Monitors Control Register

[PMEVCNTR<n>](#): Performance Monitors Event Count Registers

[PMEVTYPER<n>](#): Performance Monitors Event Type Registers

[PMINTENCLR](#): Performance Monitors Interrupt Enable Clear register

[PMINTENSET](#): Performance Monitors Interrupt Enable Set register

[PMMIR](#): Performance Monitors Machine Identification Register

[PMOVSr](#): Performance Monitors Overflow Flag Status Register

[PMOVSSET](#): Performance Monitors Overflow Flag Status Set register

[PMSELR](#): Performance Monitors Event Counter Selection Register

[PMSWINC](#): Performance Monitors Software Increment register

[PMUSERENR](#): Performance Monitors User Enable Register

[PMXEVCNTR](#): Performance Monitors Selected Event Count Register

[PMXEVTYPER](#): Performance Monitors Selected Event Type Register

PRRR: Primary Region Remap Register

REVIDR: Revision ID Register

RMR: Reset Management Register

RVBAR: Reset Vector Base Address Register

SCR: Secure Configuration Register

[SCTLR](#): System Control Register

SDCR: Secure Debug Control Register

SDER: Secure Debug Enable Register

SPSR: Saved Program Status Register

SPSR_abt: Saved Program Status Register (Abort mode)

SPSR_fiq: Saved Program Status Register (FIQ mode)

SPSR_hyp: Saved Program Status Register (Hyp mode)

SPSR_irq: Saved Program Status Register (IRQ mode)

SPSR_mon: Saved Program Status Register (Monitor mode)

SPSR_svc: Saved Program Status Register (Supervisor mode)

SPSR_und: Saved Program Status Register (Undefined mode)

TCMTR: TCM Type Register

TLBTR: TLB Type Register

TPIDRPRW: PL1 Software Thread ID Register

[TPIDRURO](#): PL0 Read-Only Software Thread ID Register

[TPIDRURW](#): PL0 Read/Write Software Thread ID Register

TRFCR: Trace Filter Control Register

TTBCR: Translation Table Base Control Register

TTBCR2: Translation Table Base Control Register 2

TTBR0: Translation Table Base Register 0

TTBR1: Translation Table Base Register 1

VBAR: Vector Base Address Register

VDFSR: Virtual SError Exception Syndrome Register

[VDISR](#): Virtual Deferred Interrupt Status Register

VMPIDR: Virtualization Multiprocessor ID Register

VPIDR: Virtualization Processor ID Register

VTCT: Virtualization Translation Control Register

VTTBR: Virtualization Translation Table Base Register

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AArch32 System Instructions

ATS12NSOPR: Address Translate Stages 1 and 2 Non-secure Only PL1 Read

ATS12NSOPW: Address Translate Stages 1 and 2 Non-secure Only PL1 Write

ATS12NSOUR: Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read

ATS12NSOUW: Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write

ATS1CPR: Address Translate Stage 1 Current state PL1 Read

ATS1CPRP: Address Translate Stage 1 Current state PL1 Read PAN

ATS1CPW: Address Translate Stage 1 Current state PL1 Write

ATS1CPWP: Address Translate Stage 1 Current state PL1 Write PAN

ATS1CUR: Address Translate Stage 1 Current state Unprivileged Read

ATS1CUW: Address Translate Stage 1 Current state Unprivileged Write

ATS1HR: Address Translate Stage 1 Hyp mode Read

ATS1HW: Address Translate Stage 1 Hyp mode Write

BPIALL: Branch Predictor Invalidate All

BPIALLIS: Branch Predictor Invalidate All, Inner Shareable

BPIMVA: Branch Predictor Invalidate by VA

[CFPRCTX](#): Control Flow Prediction Restriction by Context

[COSPRCTX](#): Clear Other Speculative Restriction by Context

CP15DMB: Data Memory Barrier System instruction

CP15DSB: Data Synchronization Barrier System instruction

CP15ISB: Instruction Synchronization Barrier System instruction

[CPPRCTX](#): Cache Prefetch Prediction Restriction by Context

DCCIMVAC: Data Cache line Clean and Invalidate by VA to PoC

DCCISW: Data Cache line Clean and Invalidate by Set/Way

DCCMVAC: Data Cache line Clean by VA to PoC

DCCMVAU: Data Cache line Clean by VA to PoU

DCCSW: Data Cache line Clean by Set/Way

DCIMVAC: Data Cache line Invalidate by VA to PoC

DCISW: Data Cache line Invalidate by Set/Way

DTLBIALL: Data TLB Invalidate All

DTLBIASID: Data TLB Invalidate by ASID match

DTLBIMVA: Data TLB Invalidate by VA

[DVPRCTX](#): Data Value Prediction Restriction by Context

ICIALLU: Instruction Cache Invalidate All to PoU

ICIALLUIS: Instruction Cache Invalidate All to PoU, Inner Shareable

ICIMVAU: Instruction Cache line Invalidate by VA to PoU

ITLBIALL: Instruction TLB Invalidate All

ITLBIASID: Instruction TLB Invalidate by ASID match

ITLBIMVA: Instruction TLB Invalidate by VA

[TLBIALL](#): TLB Invalidate All

TLBIALLH: TLB Invalidate All, Hyp mode

TLBIALLHIS: TLB Invalidate All, Hyp mode, Inner Shareable

TLBIALLIS: TLB Invalidate All, Inner Shareable

TLBIALLNSNH: TLB Invalidate All, Non-Secure Non-Hyp

TLBIALLNSNHIS: TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable

[TLBIASID](#): TLB Invalidate by ASID match

TLBIASIDIS: TLB Invalidate by ASID match, Inner Shareable

TLBIIPAS2: TLB Invalidate by Intermediate Physical Address, Stage 2

TLBIIPAS2IS: TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable

TLBIIPAS2L: TLB Invalidate by Intermediate Physical Address, Stage 2, Last level

TLBIIPAS2LIS: TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable

[TLBIMVA](#): TLB Invalidate by VA

[TLBIMVAA](#): TLB Invalidate by VA, All ASID

TLBIMVAAIS: TLB Invalidate by VA, All ASID, Inner Shareable

[TLBIMVAAL](#): TLB Invalidate by VA, All ASID, Last level

TLBIMVAALIS: TLB Invalidate by VA, All ASID, Last level, Inner Shareable

TLBIMVAH: TLB Invalidate by VA, Hyp mode

TLBIMVAHIS: TLB Invalidate by VA, Hyp mode, Inner Shareable

TLBIMVAIS: TLB Invalidate by VA, Inner Shareable

[TLBIMVAL](#): TLB Invalidate by VA, Last level

TLBIMVALH: TLB Invalidate by VA, Last level, Hyp mode

TLBIMVALHIS: TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable

TLBIMVALIS: TLB Invalidate by VA, Last level, Inner Shareable

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AMCNTENCLR0, Activity Monitors Count Enable Clear Register 0

The AMCNTENCLR0 characteristics are:

Purpose

Disable control bits for the architected activity monitors event counters, [AMEVCNTR0<n>](#).

Configuration

AArch32 System register AMCNTENCLR0 bits [31:0] are architecturally mapped to AArch64 System register [AMCNTENCLR0_ELO\[31:0\]](#).

AArch32 System register AMCNTENCLR0 bits [31:0] are architecturally mapped to External register [AMCNTENCLR0\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENCLR0 are UNDEFINED.

Attributes

AMCNTENCLR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RES0																RAZ/WI														P3	P2	P1	P0

Bits [31:16]

Reserved, RES0.

Bits [15:4]

Reserved, RAZ/WI.

This field is reserved for additional architected activity monitor event counters, which Arm might define in a future version of the Activity Monitors architecture.

P<n>, bit [n], for n = 3 to 0

Activity monitor event counter disable bit for [AMEVCNTR0<n>](#).

Note

[AMCGCR.CG0NC](#) identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4.

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR0<n> is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR0<n> is enabled. When written, disables AMEVCNTR0<n> .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENCLR0

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0010	0b100


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN0
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR0;
elsif PSTATE.EL == EL3 then
    R[t] = AMCNTENCLR0;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0010	0b100

```

if PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
    AArch32.TakeHypTrapException(0x03);
elsif IsHighestEL(PSTATE.EL) then
    AMCNTENCLR0 = R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMCNTENCLR1, Activity Monitors Count Enable Clear Register 1

The AMCNTENCLR1 characteristics are:

Purpose

Disable control bits for the auxiliary activity monitors event counters, [AMEVCNTR1<n>](#).

Configuration

AArch32 System register AMCNTENCLR1 bits [31:0] are architecturally mapped to AArch64 System register [AMCNTENCLR1_ELO\[31:0\]](#).

AArch32 System register AMCNTENCLR1 bits [31:0] are architecturally mapped to External register [AMCNTENCLR1\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENCLR1 are UNDEFINED.

Attributes

AMCNTENCLR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Bits [31:16]

Reserved, RES0.

P<n>, bit [n], for n = 15 to 0

Activity monitor event counter disable bit for [AMEVCNTR1<n>](#).

When N is less than 16, bits [15:N] are RAZ/WI, where N is the value in [AMCGCR.CG1NC](#).

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR1<n> is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR1<n> is enabled. When written, disables AMEVCNTR1<n> .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENCLR1

If the number of auxiliary activity monitor event counters implemented is zero, reads and writes of AMCNTENCLR1 are UNDEFINED.

Note

The number of auxiliary activity monitor event counters implemented is zero exactly when [AMCFGR.NCG](#) == 0b0000.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN1
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR1;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENCLR1;
elsif PSTATE.EL == EL3 then
    R[t] = AMCNTENCLR1;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0011	0b000

```

if PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
    AArch32.TakeHypTrapException(0x03);
elsif IsHighestEL(PSTATE.EL) then
    AMCNTENCLR1 = R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

AMCNTENSET0, Activity Monitors Count Enable Set Register 0

The AMCNTENSET0 characteristics are:

Purpose

Enable control bits for the architected activity monitors event counters, [AMEVCNTR0<n>](#).

Configuration

AArch32 System register AMCNTENSET0 bits [31:0] are architecturally mapped to AArch64 System register [AMCNTENSET0_ELO\[31:0\]](#).

AArch32 System register AMCNTENSET0 bits [31:0] are architecturally mapped to External register [AMCNTENSET0\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENSET0 are UNDEFINED.

Attributes

AMCNTENSET0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RES0																RAZ/WI														P3	P2	P1	P0

Bits [31:16]

Reserved, RES0.

Bits [15:4]

Reserved, RAZ/WI.

This field is reserved for additional architected activity monitor event counters, which Arm might define in a future version of the Activity Monitors architecture.

P<n>, bit [n], for n = 3 to 0

Activity monitor event counter enable bit for [AMEVCNTR0<n>](#).

Note

[AMCGCR.CG0NC](#) identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4.

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR0<n> is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR0<n> is enabled. When written, enables AMEVCNTR0<n> .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENSET0

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0010	0b101


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN0
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET0;
elsif PSTATE.EL == EL3 then
    R[t] = AMCNTENSET0;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0010	0b101

```

if PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
    AArch32.TakeHypTrapException(0x03);
elsif IsHighestEL(PSTATE.EL) then
    AMCNTENSET0 = R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The AMCNTENSET1 characteristics are:

Purpose

Enable control bits for the auxiliary activity monitors event counters, [AMEVCNTR1<n>](#).

Configuration

AArch32 System register AMCNTENSET1 bits [31:0] are architecturally mapped to AArch64 System register [AMCNTENSET1_EL0\[31:0\]](#).

AArch32 System register AMCNTENSET1 bits [31:0] are architecturally mapped to External register [AMCNTENSET1\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMCNTENSET1 are UNDEFINED.

Attributes

AMCNTENSET1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
																RES0																		P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Bits [31:16]

Reserved, RES0.

P<n>, bit [n], for n = 15 to 0

Activity monitor event counter enable bit for AMEVCNTR1<n>.

When N is less than 16, bits [15:N] are RAZ/WI, where N is the value in [AMCGCR.CG1NC](#).

Possible values of each bit are:

P<n>	Meaning
0b0	When read, means that AMEVCNTR1<n> is disabled. When written, has no effect.
0b1	When read, means that AMEVCNTR1<n> is enabled. When written, enables AMEVCNTR1<n> .

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMCNTENSET1

If the number of auxiliary activity monitor event counters implemented is zero, reads and writes of AMCNTENSET1 are UNDEFINED.

Note

The number of auxiliary activity monitor counters implemented is zero when [AMCFGR.NCG](#) == 0b0000.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0011	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HAFGRTR_EL2.AMCNTEN1
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET1;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMCNTENSET1;
elsif PSTATE.EL == EL3 then
    R[t] = AMCNTENSET1;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0011	0b001

```

if PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
    AArch32.TakeHypTrapException(0x03);
elsif IsHighestEL(PSTATE.EL) then
    AMCNTENSET1 = R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMEVCNTR0<n>, Activity Monitors Event Counter Registers 0, n = 0 - 3

The AMEVCNTR0<n> characteristics are:

Purpose

Provides access to the architected activity monitor event counters.

Configuration

AArch32 System register AMEVCNTR0<n> bits [63:0] are architecturally mapped to AArch64 System register [AMEVCNTR0<n>_EL0\[63:0\]](#).

AArch32 System register AMEVCNTR0<n> bits [63:0] are architecturally mapped to External register [AMEVCNTR0<n>\[63:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVCNTR0<n> are UNDEFINED.

Attributes

AMEVCNTR0<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ACNT															
																ACNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ACNT, bits [63:0]

Architected activity monitor event counter n.

Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.

If FEAT_AMUv1p1 is implemented, [HCR_EL2](#).AMVOFFEN is 1, [SCR_EL3](#).AMVOFFEN is 1, [HCR_EL2](#).{E2H, TGE} is not {1,1}, and EL2 is using AArch64 and is implemented in the current Security state, access to these registers at EL0 or EL1 return (PCount<63:0> - [AMEVCNTVOFF0<n>_EL2](#)<63:0>).

PCount is the physical count returned when AMEVCNTR0<n> is read from EL2 or EL3.

If the counter is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMEVCNTR0<n>

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n> are UNDEFINED.

Note

[AMCGCR](#).CG0NC identifies the number of architected activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm> ; Where m = 0-3

coproc	CRm	opc1
0b1111	0b000:m[3]	0b0:m[2:0]


```

integer m = UInt(CRm<0>:opc1<2:0>);

if m >= 4 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
    elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && m < 8 &&
HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && m < 8 && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HAFGRTR_EL2.AMEVCNTR0<m>_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (AMEVCNTR0[m]<63:32>, AMEVCNTR0[m]<31:0>);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && m < 8 && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && m < 8 && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (AMEVCNTR0[m]<63:32>, AMEVCNTR0[m]<31:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (AMEVCNTR0[m]<63:32>, AMEVCNTR0[m]<31:0>);

```

```

elseif PSTATE.EL == EL3 then
    (R[t2], R[t]) = (AMEVCNTR0[m]<63:32>, AMEVCNTR0[m]<31:0>);

```

MCRR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm> ; Where m = 0-3

coproc	CRm	opc1
0b1111	0b000:m[3]	0b0:m[2:0]

```

integer m = UInt(CRm<0>:opc1<2:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && m < 8 && HSTR_EL2.T0 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x04);
elseif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && m < 8 && HSTR.T0 == '1' then
    AArch32.TakeHypTrapException(0x04);
elseif IsHighestEL(PSTATE.EL) then
    AMEVCNTR0[m] = R[t2]:R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMEVCNTR1<n>, Activity Monitors Event Counter Registers 1, n = 0 - 15

The AMEVCNTR1<n> characteristics are:

Purpose

Provides access to the auxiliary activity monitor event counters.

Configuration

AArch32 System register AMEVCNTR1<n> bits [63:0] are architecturally mapped to AArch64 System register [AMEVCNTR1<n>_EL0\[63:0\]](#).

AArch32 System register AMEVCNTR1<n> bits [63:0] are architecturally mapped to External register [AMEVCNTR1<n>\[63:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVCNTR1<n> are UNDEFINED.

Attributes

AMEVCNTR1<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ACNT															
																ACNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ACNT, bits [63:0]

Auxiliary activity monitor event counter n.

Value of auxiliary activity monitor event counter n, where n is the number of this register and is a number from 0 to 15.

If FEAT_AMUv1p1 is implemented, [HCR_EL2](#).AMVOFFEN is 1, [SCR_EL3](#).AMVOFFEN is 1, [HCR_EL2](#).{E2H, TGE} is not {1,1}, EL2 is using AArch64 and is implemented in the current Security state, and [AMCR_EL0](#).CG1RZ is 0, reads to these registers at EL0 or EL1 return (PCount<63:0> - [AMEVCNTVOFF1<n>_EL2<63:0>](#)).

PCount is the physical count returned when AMEVCNTR1<n> is read from EL2 or EL3.

If the counter is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On an AMU reset, this field resets to 0.

Accessing AMEVCNTR1<n>

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n> are UNDEFINED.

Note

[AMCGCR](#).CG1NC identifies the number of auxiliary activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm> ; Where m = 0-15

coproc	CRm	opc1
0b1111	0b010:m[3]	0b0:m[2:0]

```

integer m = UInt(CRm<0>:opc1<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
        elsif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x04);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && m >= 8 &&
HSTR_EL2.T5 == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x04);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && m >= 8 && HSTR.T5 == '1' then
                AArch32.TakeHypTrapException(0x04);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x04);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
                AArch32.TakeHypTrapException(0x04);
            elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HAFGRTR_EL2.AMEVCNTR1<m>_EL0 == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x04);
            elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.AArch32SystemAccessTrap(EL3, 0x04);
                elsif HaveAArch64() && AMCR_EL0.CG1RZ == '1' then
                    (R[t2], R[t]) = (Zeros(32), Zeros(32));
                elsif !HaveAArch64() && AMCR.CG1RZ == '1' then
                    (R[t2], R[t]) = (Zeros(32), Zeros(32));
                else
                    (R[t2], R[t]) = (AMEVCNTR1[m]<63:32>, AMEVCNTR1[m]<31:0>);
            elsif PSTATE.EL == EL1 then
                if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
                    UNDEFINED;
                elsif EL2Enabled() && !ELUsingAArch32(EL2) && m >= 8 && HSTR_EL2.T5 == '1' then
                    AArch64.AArch32SystemAccessTrap(EL2, 0x04);
                elsif EL2Enabled() && ELUsingAArch32(EL2) && m >= 8 && HSTR.T5 == '1' then
                    AArch32.TakeHypTrapException(0x04);
                elsif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
                    AArch64.AArch32SystemAccessTrap(EL2, 0x04);
                elsif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
                    AArch32.TakeHypTrapException(0x04);
                elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
                    if Halted() && EDSCR.SDD == '1' then
                        UNDEFINED;
                    else
                        AArch64.AArch32SystemAccessTrap(EL3, 0x04);
                    elsif !IsHighestEL(PSTATE.EL) && HaveAArch64() && AMCR_EL0.CG1RZ == '1' then
                        (R[t2], R[t]) = (Zeros(32), Zeros(32));
                    elsif !IsHighestEL(PSTATE.EL) && !HaveAArch64() && AMCR.CG1RZ == '1' then
                        (R[t2], R[t]) = (Zeros(32), Zeros(32));
                    else
                        (R[t2], R[t]) = (AMEVCNTR1[m]<63:32>, AMEVCNTR1[m]<31:0>);
                elsif PSTATE.EL == EL2 then

```

```

    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
        elsif !IsHighestEL(PSTATE.EL) && HaveAArch64() && AMCR_EL0.CG1RZ == '1' then
            (R[t2], R[t]) = (Zeros(32), Zeros(32));
        elsif !IsHighestEL(PSTATE.EL) && !HaveAArch64() && AMCR.CG1RZ == '1' then
            (R[t2], R[t]) = (Zeros(32), Zeros(32));
        else
            (R[t2], R[t]) = (AMEVCNTR1[m]<63:32>, AMEVCNTR1[m]<31:0>);
    elsif PSTATE.EL == EL3 then
        (R[t2], R[t]) = (AMEVCNTR1[m]<63:32>, AMEVCNTR1[m]<31:0>);

```

MCRR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm> ; Where m = 0-15

coproc	CRm	opc1
0b1111	0b010:m[3]	0b0:m[2:0]

```

integer m = UInt(CRm<0>:opc1<2:0>);

if m >= NUM_AMU.CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && m >= 8 && HSTR_EL2.T5 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x04);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && m >= 8 && HSTR.T5 == '1' then
    AArch32.TakeHypTrapException(0x04);
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR1[m] = R[t2]:R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AMEVTYPER1<n>, Activity Monitors Event Type Registers 1, n = 0 - 15

The AMEVTYPER1<n> characteristics are:

Purpose

Provides information on the events that an auxiliary activity monitor event counter [AMEVCNTR1<n>](#) counts.

Configuration

AArch32 System register AMEVTYPER1<n> bits [31:0] are architecturally mapped to AArch64 System register [AMEVTYPER1<n>_EL0\[31:0\]](#).

AArch32 System register AMEVTYPER1<n> bits [31:0] are architecturally mapped to External register [AMEVTYPER1<n>\[31:0\]](#).

This register is present only when FEAT_AMUv1 is implemented. Otherwise, direct accesses to AMEVTYPER1<n> are UNDEFINED.

Attributes

AMEVTYPER1<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																evtCount															

Bits [31:16]

Reserved, RES0.

evtCount, bits [15:0]

Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter [AMEVCNTR1<n>](#).

It is IMPLEMENTATION DEFINED what values are supported by each counter.

If software writes a value to this field which is not supported by the corresponding counter [AMEVCNTR1<n>](#), then:

- It is UNPREDICTABLE which event will be counted.
- The value read back is UNKNOWN.

The event counted by [AMEVCNTR1<n>](#) might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.

If the corresponding counter [AMEVCNTR1<n>](#) is enabled, writes to this register have UNPREDICTABLE results.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing AMEVTYPER1<n>

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n> are UNDEFINED.

Note

[AMCGCR.CG1NC](#) identifies the number of auxiliary activity monitor event counters.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b111:m[3]	m[2:0]


```

integer m = UInt(CRm<0>:opc2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elseif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elseif !ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HAFGRTR_EL2.AMEVTYPER1<m>_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMEVTYPER1[m];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && CPTR_EL2.TAM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCPTR.TAM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = AMEVTYPER1[m];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);

```

```

else
    R[t] = AMEVTYPER1[m];
elsif PSTATE.EL == EL3 then
    R[t] = AMEVTYPER1[m];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b111:m[3]	m[2:0]

```

integer m = UInt(CRm<0>:opc2<2:0>);

if m >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif !IsG1ActivityMonitorImplemented(m) then
    UNDEFINED;
elsif PSTATE.EL == EL1 && EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
elsif PSTATE.EL == EL1 && EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
    AArch32.TakeHypTrapException(0x03);
elsif IsHighestEL(PSTATE.EL) && !boolean IMPLEMENTATION_DEFINED "AMEVCNTR1[m] is fixed" then
    AMEVTYPER1[m] = R[t];
else
    UNDEFINED;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CFPRCTX, Control Flow Prediction Restriction by Context

The CFPRCTX characteristics are:

Purpose

Control Flow Prediction Restriction by Context applies to all Control Flow Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Control flow predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when AArch32 is supported and FEAT_SPECRES is implemented. Otherwise, direct accesses to CFPRCTX are UNDEFINED.

Attributes

CFPRCTX is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				GVMIDNS				EL				VMID				RES0				GASID				ASID							

Bits [31:28]

Reserved, RES0.

GVMID, bit [27]

Execution of this instruction applies to all VMIDs or a specified VMID.

GV MID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

NS, bit [26]

Security State.

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

If the instruction is executed in Non-secure state, this field has an Effective value of 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

VMID, bits [23:16]

Only applies when bit[27] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)) or EL2 is using AArch32 state.

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#) or [ELUsingAArch32\(EL2\)](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#) and [!ELUsingAArch32\(EL2\)](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

Bits [15:9]

Reserved, RES0.

GASID, bit [8]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field is treated as 0.

ASID, bits [7:0]

Only applies for an EL0 target execution context and when bit[8] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

Executing the CFPRCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0111	0b0011	0b100

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.EnRCTX ==
'0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && SCTL_EL1.EnRCTX == '0' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T7 == '1'
then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
                AArch32.TakeHypTrapException(0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CFPRCTX ==
'1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.EnRCTX ==
'0' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            else
                AArch32.RestrictPrediction(R[t], RestrictType_ControlFlow);
        elsif PSTATE.EL == EL1 then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
                AArch32.TakeHypTrapException(0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.NV == '1' then
                AArch64.SystemAccessTrap(EL2, 0x03);
            else
                AArch32.RestrictPrediction(R[t], RestrictType_ControlFlow);
        elsif PSTATE.EL == EL2 then
            AArch32.RestrictPrediction(R[t], RestrictType_ControlFlow);
        elsif PSTATE.EL == EL3 then
            AArch32.RestrictPrediction(R[t], RestrictType_ControlFlow);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CLIDR, Cache Level ID Register

The CLIDR characteristics are:

Purpose

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configuration

AArch32 System register CLIDR bits [31:0] are architecturally mapped to AArch64 System register [CLIDR_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to CLIDR are UNDEFINED.

Attributes

CLIDR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICB		LoUU		LoC		LoUIS		Ctype7		Ctype6		Ctype5		Ctype4		Ctype3		Ctype2		Ctype1											

ICB, bits [31:30]

Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.

ICB	Meaning
0b00	Not disclosed by this mechanism.
0b01	L1 cache is the highest Inner Cacheable level.
0b10	L2 cache is the highest Inner Cacheable level.
0b11	L3 cache is the highest Inner Cacheable level.

LoUU, bits [29:27]

Level of Unification Uniprocessor for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

Note

This field does not describe the requirements for instruction cache invalidation. See [CTR.DIC](#).

Note

When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches.

LoC, bits [26:24]

Level of Coherence for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

LoUIS, bits [23:21]

Level of Unification Inner Shareable for the cache hierarchy.

For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.

Note

This field does not describe the requirements for instruction cache invalidation. See [CTR.DIC](#).

Note

When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches.

Ctype<n>, bits [3(n-1)+2:3(n-1)], for n = 7 to 1

Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.

Ctype<n>	Meaning
0b000	No cache.
0b001	Instruction cache only.
0b010	Data cache only.
0b011	Separate instruction and data caches.
0b100	Unified cache.

All other values are reserved.

If software reads the Cache Type fields from Ctype1 upwards, once it has seen a value of 000, no caches that can be managed using the architected cache maintenance instructions that operate by set/way exist at further-out levels of the hierarchy. So, for example, if Ctype3 is the first Cache Type field with a value of 000, the values of Ctype4 to Ctype7 must be ignored.

Accessing CLIDR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b001	0b0000	0b0000	0b001


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TID2 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TID4 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TID2 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR2.TID4 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        R[t] = CLIDR;
    end
elsif PSTATE.EL == EL2 then
    R[t] = CLIDR;
elsif PSTATE.EL == EL3 then
    R[t] = CLIDR;

```

3095/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

COSPRCTX, Clear Other Speculative Restriction by Context

The COSPRCTX characteristics are:

Purpose

Clear Other Speculative Prediction Restriction by Context applies to prediction resources not managed by another of the speculation restriction System instructions.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively any predictions occurring after the instruction is complete and synchronized.

This instruction applies to all speculative access except:

- Cache Prefetch predictions.
- Data Value predictions.
- Control Flow predictions.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when AArch32 is supported and FEAT_SPECRES2 is implemented. Otherwise, direct accesses to COSPRCTX are UNDEFINED.

Attributes

COSPRCTX is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				GVMIDNS				EL				VMID				RES0				GASID		ASID									

Bits [31:28]

Reserved, RES0.

GVMID, bit [27]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMD	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, then this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

NS, bit [26]

Security State.

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

If the instruction is executed in Non-secure state, this field is treated as 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

VMID, bits [23:16]

Only applies when bit[27] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)) or EL2 is using AArch32 state.

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#) or [ELUsingAArch32\(EL2\)](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#) and [!ELUsingAArch32\(EL2\)](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

Bits [15:9]

Reserved, RES0.

GASID, bit [8]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [7:0]

Only applies for an EL0 target execution context and when bit[8] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

Executing COSPRCTX

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0111	0b0011	0b110

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.EnRCTX ==
'0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && SCTL_EL1.EnRCTX == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T7 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.COSPRCTX ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.EnRCTX ==
'0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    else
        AArch32.RestrictPrediction(R[t], RestrictType_Other);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x03);
    else
        AArch32.RestrictPrediction(R[t], RestrictType_Other);
elseif PSTATE.EL == EL2 then
    AArch32.RestrictPrediction(R[t], RestrictType_Other);
elseif PSTATE.EL == EL3 then
    AArch32.RestrictPrediction(R[t], RestrictType_Other);

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPPRCTX, Cache Prefetch Prediction Restriction by Context

The CPPRCTX characteristics are:

Purpose

Cache Prefetch Prediction Restriction by Context applies to all Cache Allocation Resources that predict cache allocations based on information gathered within the target execution context or contexts.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control cache prefetch predictions occurring after the instruction is complete and synchronized.

This instruction applies to all:

- Instruction caches.
- Data caches.
- TLB prefetching hardware used by the executing PE that applies to the supplied context or contexts.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when AArch32 is supported and FEAT_SPECRES is implemented. Otherwise, direct accesses to CPPRCTX are UNDEFINED.

Attributes

CPPRCTX is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				GVMIDNS				EL				VMID				RES0				GASID				ASID							

Bits [31:28]

Reserved, RES0.

GVMID, bit [27]

Execution of this instruction applies to all VMIDs or a specified VMID.

GVMID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, then this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

NS, bit [26]

Security State.

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

If the instruction is executed in Non-secure state, this field is treated as 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

VMID, bits [23:16]

Only applies when bit[27] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)) or EL2 is using AArch32 state.

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#) or [ELUsingAArch32\(EL2\)](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#) and [!ELUsingAArch32\(EL2\)](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

Bits [15:9]

Reserved, RES0.

GASID, bit [8]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [7:0]

Only applies for an EL0 target execution context and when bit[8] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

Executing the CPPRCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0111	0b0011	0b111


```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_EL1.EnRCTX ==
'0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && SCTL_EL1.EnRCTX == '0' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T7 == '1'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX ==
'1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.EnRCTX ==
'0' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch32.RestrictPrediction(R[t], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x03);
        else
            AArch32.RestrictPrediction(R[t], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL2 then
        AArch32.RestrictPrediction(R[t], RestrictType_CachePrefetch);
    elsif PSTATE.EL == EL3 then
        AArch32.RestrictPrediction(R[t], RestrictType_CachePrefetch);

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGBCR<n>, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n> characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register [DBGBVR<n>](#). If EL2 is implemented and this breakpoint supports Context matching, [DBGBVR<n>](#) can be associated with a Breakpoint Extended Value Register [DBGBXVR<n>](#) for VMID matching.

Configuration

AArch32 System register DBGBCR<n> bits [31:0] are architecturally mapped to AArch64 System register [DBGBCR<n>_EL1\[31:0\]](#).

AArch32 System register DBGBCR<n> bits [31:0] are architecturally mapped to External register [DBGBCR<n>_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to DBGBCR<n> are UNDEFINED.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

DBGBCR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								BT				LBN				SSC		HMC	RES0				BAS			RES0	PMC	E			

When the E field is zero, all the other fields in the register are ignored.

Bits [31:24]

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

BT	Meaning
0b0000	Unlinked instruction address match. DBGBVR<n> is the address of an instruction.
0b0001	As 0b0000, but linked to a Context matching breakpoint.
0b0010	Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of HCR_EL2.E2H is 1, if either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>.ContextID must match the CONTEXTIDR value.
0b0011	As 0b0010 with linking enabled.
0b0100	Unlinked instruction address mismatch. DBGBVR<n> is the address of an instruction to be stepped.
0b0101	As 0b0100, but linked to a Context matching breakpoint.
0b0110	Unlinked CONTEXTIDR_EL1 match. DBGBVR<n>.ContextID is a Context ID compared against CONTEXTIDR .
0b0111	As 0b0110 with linking enabled.
0b1000	Unlinked VMID match. DBGBXVR<n>.VMID is a VMID compared against VTTBR.VMID .
0b1001	As 0b1000 with linking enabled.
0b1010	Unlinked VMID and Context ID match. DBGBVR<n>.ContextID is a Context ID compared against CONTEXTIDR , and DBGBXVR<n>.VMID is a VMID compared against VTTBR.VMID .
0b1011	As 0b1010 with linking enabled.
0b1100	Unlinked CONTEXTIDR_EL2 match. DBGBXVR<n>.ContextID2 is a Context ID compared against CONTEXTIDR_EL2 .
0b1101	As 0b1100 with linking enabled.
0b1110	Unlinked Full Context ID match. DBGBVR<n>.ContextID is compared against CONTEXTIDR , and DBGBXVR<n>.ContextID2 is compared against CONTEXTIDR_EL2 .
0b1111	As 0b1110 with linking enabled.

For more information on Breakpoints and their constraints, see 'Breakpoint exceptions' and 'Reserved DBGBCR<n>.BT values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked [Breakpoint breakpoint Number number](#). For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For all other breakpoint types, this field is ignored and reads of the register return an UNKNOWN value.

This field is ignored when the value of DBGBCR<n>.E is 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields.

For more information, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' and 'Reserved DBGBCR<n>.{SSC, HMC, PMC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the SSC, bits [15:14] description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0b0011	DBGBVR<n>	Use for T32 instructions
0b1100	DBGBVR<n> +2	Use for T32 instructions
0b1111	DBGBVR<n>	Use for A32 instructions

All other values are reserved. For more information, see 'Reserved DBGBCR<n>.BAS values'.

For more information on using the BAS field in Address Match breakpoints, see 'Using the BAS field in Address Match breakpoints'.

For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are:

BAS	Step instruction at	Constraint for debuggers
0b0000	-	Use for a match anywhere breakpoint
0b0011	DBGBVR<n>	Use for T32 instructions
0b1100	DBGBVR<n> +2	Use for T32 instructions
0b1111	DBGBVR<n>	Use for A32 instructions

All other values are reserved. For more information, see 'Reserved DBGBCR<n>.BAS values'.

For more information on using the BAS field in address mismatch breakpoints, see 'Using the BAS field in Address Match breakpoints'.

For Context matching breakpoints, this field is RES1 and ignored.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [4:3]

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>.SSC description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable breakpoint [DBGBVR<n>](#). Possible values are:

E	Meaning
0b0	Breakpoint disabled.
0b1	Breakpoint enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing DBGBCR<n>

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0000	m[3:0]	0b101

```

integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        R[t] = DBGBCR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        R[t] = DBGBCR[m];
elsif PSTATE.EL == EL3 then
    if DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        R[t] = DBGBCR[m];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0000	m[3:0]	0b101

```
integer m = UInt(CRm<3:0>);

if m >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR[m] = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR[m] = R[t];
elsif PSTATE.EL == EL3 then
    if DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR[m] = R[t];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGDEVID, Debug Device ID register 0

The DBGDEVID characteristics are:

Purpose

Adds to the information given by the [DBGDIDR](#) by describing other features of the debug implementation.

Configuration

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to DBGDEVID are UNDEFINED.

This register is required in all implementations.

Attributes

DBGDEVID is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDMask				AuxRegs				DoubleLock				VirtExtns				VectorCatch				BPAAddrMask				WPAddrMask				PCSample			

CIDMask, bits [31:28]

Indicates the level of support for the Context ID matching breakpoint masking capability. Defined values are:

CIDMask	Meaning
0b0000	Context ID masking is not implemented.
0b0001	Context ID masking is implemented.

All other values are reserved. The value of this for Armv8 is 0b0000.

AuxRegs, bits [27:24]

Indicates support for Auxiliary registers. Permitted values for this field are:

AuxRegs	Meaning
0b0000	None supported.
0b0001	Support for External Debug Auxiliary Control Register, EDACR .

All other values are reserved.

DoubleLock, bits [23:20]

OS Double Lock implemented. Defined values are:

DoubleLock	Meaning
0b0000	OS Double Lock is not implemented. DBGOSDLR is RAZ/WI.
0b0001	OS Double Lock is implemented. DBGOSDLR is RW.

FEAT_DoubleLock implements the functionality identified by the value 0b0001.

All other values are reserved.

VirtExtns, bits [19:16]

Indicates whether EL2 is implemented. Defined values are:

VirtExtns	Meaning
0b0000	EL2 is not implemented.
0b0001	EL2 is implemented.

All other values are reserved.

VectorCatch, bits [15:12]

Defines the form of Vector Catch exception implemented. Defined values are:

VectorCatch	Meaning
0b0000	Address matching Vector Catch exception implemented.
0b0001	Exception matching Vector Catch exception implemented.

All other values are reserved.

BPAAddrMask, bits [11:8]

Indicates the level of support for the instruction address matching breakpoint masking capability. Defined values are:

BPAAddrMask	Meaning
0b0000	Breakpoint address masking might be implemented. If not implemented, DBGBCR<n> [28:24] is RAZ/WI.
0b0001	Breakpoint address masking is implemented.
0b1111	Breakpoint address masking is not implemented. DBGBCR<n> [28:24] is RES0.

All other values are reserved. The value of this for Armv8 is 0b1111.

WPAAddrMask, bits [7:4]

Indicates the level of support for the data address matching watchpoint masking capability. Defined values are:

WPAAddrMask	Meaning
0b0000	Watchpoint address masking might be implemented. If not implemented, DBGWCR<n>.MASK (Address mask) is RAZ/WI.
0b0001	Watchpoint address masking is implemented.
0b1111	Watchpoint address masking is not implemented. DBGWCR<n>.MASK (Address mask) is RES0.

All other values are reserved. The value of this for Armv8 is 0b0001.

PCSample, bits [3:0]

Indicates the level of PC Sample-based Profiling support using external debug registers. Defined values are:

PCSample	Meaning
0b0000	PC Sample-based Profiling Extension is not implemented in the external debug registers space.
0b0010	Only EDPCSR and EDCIDS are implemented. This option is only permitted if EL3 and EL2 are not implemented.
0b0011	EDPCSR , EDCIDS , and EDVIDSR are implemented.

All other values are reserved.

When FEAT_PCSRv8p2 is implemented, the only permitted value is 0b0000.

Note

FEAT_PCSRv8p2 implements the PC Sample-based Profiling Extension in the Performance Monitors register space, as indicated by the value of **PMU.PMDEVID.PCSample**. ~~PMDEVID.PCSample~~.

Accessing DBGDEVID

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0111	0b0010	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    else
        R[t] = DBGDEVID;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    else
        R[t] = DBGDEVID;
elseif PSTATE.EL == EL3 then
    R[t] = DBGDEVID;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0111	0b0001	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
        else
            R[t] = DBGDEVID1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
        else
            R[t] = DBGDEVID1;
elsif PSTATE.EL == EL3 then
    R[t] = DBGDEVID1;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

DBGDIDR, Debug ID Register

The DBGDIDR characteristics are:

Purpose

Specifies which version of the Debug architecture is implemented, and some features of the debug implementation.

Configuration

This register is present only when AArch32 is supported. Otherwise, direct accesses to DBGDIDR are UNDEFINED.
If EL1 cannot use AArch32 then the implementation of this register is OPTIONAL and deprecated.

Attributes

DBGDIDR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPs				BRPs				CTX_CMPs				Version				RES1				RES0				SE_imp				RES0			

WRPs, bits [31:28]

NumberThe number of watchpoints implemented, minus 1.
IfPermitted values of this field are from FEAT_Debugv8p90b0001 isfor 2-implemented and 16 or more watchpoints are implemented, this field reads asto 0b1111for 16 implemented watchpoints.
The value of 0b0000 is reserved.
If AArch64 is implemented, this field has the same value as ID_AA64DFR0_EL1.WRPs.

BRPs, bits [27:24]

NumberThe number of breakpoints implemented, minus 1.
IfPermitted values of this field are from FEAT_Debugv8p90b0001 isfor 2-implemented and 16 or more breakpoints are implementedbreakpoint, this field reads asto 0b1111for 16 implemented breakpoints.
The value of 0b0000 is reserved.
If AArch64 is implemented, this field has the same value as ID_AA64DFR0_EL1.BRPs.

CTX_CMPs, bits [23:20]

The Context matching breakpoints must be the highest addressed breakpoints. For example, if six breakpoints are implemented and two are Context matching breakpoints, they must be breakpoints 4 and 5.
If AArch64 is implemented, this field has the same value as ID_AA64DFR0_EL1.CTX_CMPs.
NumberThe number of breakpoints that arecan context-awarebe used for Context matching, minus 1.

If Permitted values of this field are from FEAT_Debugv8p9 0b0000 is for implemented 1 and Context 16 matching or more breakpoints that are context-aware are implemented breakpoint, this field reads as to 0b1111 for 16 Context matching breakpoints.

Version, bits [19:16]

Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are:

Version	Meaning
0b0000	Not supported.
0b0001	Armv6, v6 Debug architecture, with System registers access.
0b0010	Armv6, v6.1 Debug architecture, with System registers access.
0b0011	Armv7, v7 Debug architecture, with only baseline System registers.
0b0100	Armv7, v7 Debug architecture, with all System registers implemented.
0b0101	Armv7, v7.1 Debug architecture, with System registers access.
0b0110	Armv8 debug architecture.
0b0111	Armv8 debug architecture with Virtualization Host Extensions.
0b1000	Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.
0b1011	Armv8.9 debug architecture, FEAT_Debugv8p9.

All other values are reserved.

The values 0b0000, 0b0001, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8.

FEAT_VHE adds the functionality identified by the value 0b0111.

FEAT_Debugv8p2 adds the functionality identified by the value 0b1000.

FEAT_Debugv8p4 adds the functionality identified by the value 0b1001.

FEAT_Debugv8p8 adds the functionality identified by the value 0b1010.

FEAT_Debugv8p9 adds the functionality identified by the value 0b1011.

From Armv8.1, when FEAT_VHE is implemented the value 0b0110 is not permitted.

From Armv8.2, the values 0b0110 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 is not permitted.

From Armv8.9, the value 0b1010 is not permitted.

Bit [15]

Reserved, RES1.

nSUHD_imp, bit [14]

Previously indicated that Secure User Halting Debug is not implemented.

The value of this bit must match the value of the SE_imp bit.

Bit [13]

Reserved, RES0.

SE_imp, bit [12]

EL3 implemented. The meanings of the values of this bit are:

SE_imp	Meaning
0b0	EL3 not implemented.
0b1	EL3 implemented.

The value of this bit must match the value of the nSUHD_imp bit.

Bits [11:0]

Reserved, RES0.

Accessing DBGDIDR

Arm deprecates any access to this register from EL0.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0000	0b0000	0b000

```

if Halted() && ConstrainUnpredictableBool(Unpredictable_IGNORETRAPINDEBUG) then
    R[t] = DBGDIDR;
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && MDSCR_EL1.TDCC == '1' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x05);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x05);
        elsif ELUsingAArch32(EL1) && DBGDSCRExt.UCCdis == '1' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x05);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && (HCR_EL2.TGE == '1' || MDCR_EL2.<TDE,TDA> !=
'00') then
                AArch64.AArch32SystemAccessTrap(EL2, 0x05);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && (HCR.TGE == '1' || HDCR.<TDE,TDA> != '00') then
                AArch32.TakeHypTrapException(0x05);
            elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.AArch32SystemAccessTrap(EL3, 0x05);
                else
                    R[t] = DBGDIDR;
            elsif PSTATE.EL == EL1 then
                if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
                    UNDEFINED;
                elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
                    AArch64.AArch32SystemAccessTrap(EL2, 0x05);
                elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
                    AArch32.TakeHypTrapException(0x05);
                elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
                    if Halted() && EDSCR.SDD == '1' then
                        UNDEFINED;
                    else
                        AArch64.AArch32SystemAccessTrap(EL3, 0x05);
                    else
                        R[t] = DBGDIDR;
            elsif PSTATE.EL == EL2 then
                if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
                    UNDEFINED;
                elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
                    if Halted() && EDSCR.SDD == '1' then
                        UNDEFINED;
                    else
                        AArch64.AArch32SystemAccessTrap(EL3, 0x05);
                    else
                        R[t] = DBGDIDR;
            elsif PSTATE.EL == EL3 then
                R[t] = DBGDIDR;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGWCR<n>, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n> characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register [DBGWVR<n>](#).

Configuration

AArch32 System register DBGWCR<n> bits [31:0] are architecturally mapped to AArch64 System register [DBGWCR<n>_EL1\[31:0\]](#).

AArch32 System register DBGWCR<n> bits [31:0] are architecturally mapped to External register [DBGWCR<n>_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to DBGWCR<n> are UNDEFINED.

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

DBGWCR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															

When the E field is zero, all the other fields in the register are ignored.

Bits [31:29]

Reserved, RES0.

MASK, bits [28:24]

Address [Mask](#).mask. Only objects up to 2GB can be watched using a single mask.

MASK	Meaning
0b00000	No mask.
0b00001	Reserved.
0b00010	Reserved.

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0b0	Unlinked data address match.
0b1	Linked data address match.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked ~~Breakpoint~~ ~~breakpoint~~ ~~Number~~ ~~number~~. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

For more information, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions', and 'Reserved DBGWCR<n>.{SSC, HMC, PAC} values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by [DBGWVR<n>](#) is being watched.

BAS	Description
0bxxxxxxx1	Match byte at DBGWVR<n>
0bxxxxxx1x	Match byte at DBGWVR<n>+1
0bxxxxx1xx	Match byte at DBGWVR<n>+2
0bxxxx1xxx	Match byte at DBGWVR<n>+3

In cases where [DBGWVR<n>](#) addresses a double-word:

BAS	Description, if DBGWVR<n> [2] == 0
0bxxx1xxxx	Match byte at DBGWVR<n>+4
0bxx1xxxxx	Match byte at DBGWVR<n>+5
0bx1xxxxxx	Match byte at DBGWVR<n>+6
0b1xxxxxxx	Match byte at DBGWVR<n>+7

If [DBGWVR<n>](#)[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting [DBGWVR<n>](#)[2] == 1.

The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>.BAS values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
0b01	Match instructions that load from a watchpointed address.
0b10	Match instructions that store to a watchpointed address.
0b11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Enable watchpoint n. Possible values are:

E	Meaning
0b0	Watchpoint disabled.
0b1	Watchpoint enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing DBGWCR<n>

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0000	m[3:0]	0b111

```
integer m = UInt(CRm<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
        elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            R[t] = DBGWCR[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x05);
        elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            R[t] = DBGWCR[m];
    elsif PSTATE.EL == EL3 then
        if DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            R[t] = DBGWCR[m];
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-15

coproc	opc1	CRn	CRm	opc2
0b1110	0b000	0b0000	m[3:0]	0b111

```

integer m = UInt(CRm<3:0>);

if m >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR[m] = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
    elsif DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR[m] = R[t];
elsif PSTATE.EL == EL3 then
    if DBGOSLSR.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR[m] = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

DSPSR2, Debug Saved Process State Register 2

The DSPSR2 characteristics are:

Purpose

Holds the saved process state for Debug state. On entering Debug state, PSTATE information is written to this register. On exiting Debug state, values are copied from this register to PSTATE.

Configuration

AArch32 System register DSPSR2 bits [31:0] are architecturally mapped to AArch64 System register [DSPSR_EL0\[63:32\]](#).

This register is present only when FEAT_Debugv8p9 is implemented. Otherwise, direct accesses to DSPSR2 are UNDEFINED.

Attributes

DSPSR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																PPEND		PM													

Bits [31:2]

Reserved, RES0.

PPEND, bit [1]

When FEAT_SEBEP is implemented:

PMU exception pending. Set to the value of PSTATE.PPEND on entering Debug state, and copied to PSTATE.PPEND on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PM, bit [0]

When FEAT_EBEP is implemented:

PMU exception mask. Set to the value of PSTATE.PM on entering Debug state, and copied to PSTATE.PM on exiting Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing DPSR2

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b011	0b0100	0b0101	0b010

```

if !Halted() then
    UNDEFINED;
else
    R[t] = DPSR2;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b011	0b0100	0b0101	0b010

```

if !Halted() then
    UNDEFINED;
else
    DPSR2 = R[t];

```

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DVPRCTX, Data Value Prediction Restriction by Context

The DVPRCTX characteristics are:

Purpose

Data Value Prediction Restriction by Context applies to all Data Value Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Data value predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.

Note

This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configuration

This instruction is present only when AArch32 is supported and FEAT_SPECRES is implemented. Otherwise, direct accesses to DVPRCTX are UNDEFINED.

Attributes

DVPRCTX is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				GVMIDNS				EL				VMID				RES0				GASID				ASID							

Bits [31:28]

Reserved, RES0.

GVMID, bit [27]

Execution of this instruction applies to all VMIDs or a specified VMID.

GV MID	Meaning
0b0	Applies to specified VMID for an EL0 or EL1 target execution context.
0b1	Applies to all VMIDs for an EL0 or EL1 target execution context.

For target execution contexts other than EL0 or EL1, this field is RES0.

If the instruction is executed at EL0 or EL1, this field has an Effective value of 0.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

NS, bit [26]

Security State.

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

If the instruction is executed in Non-secure state, this field has an Effective value of 1.

EL, bits [25:24]

Exception Level. Indicates the Exception level of the target execution context.

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

If the instruction is executed at an Exception level lower than the specified level, this instruction is treated as a NOP.

VMID, bits [23:16]

Only applies when bit[27] is 0 and the target execution context is either:

- EL1.
- EL0 when ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#)) or EL2 is using AArch32 state.

Otherwise this field is RES0.

When the instruction is executed at EL1, this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==0](#) or [HCR_EL2.TGE==0](#) or [ELUsingAArch32\(EL2\)](#)), this field is treated as the current VMID.

When the instruction is executed at EL0 and ([HCR_EL2.E2H==1](#) and [HCR_EL2.TGE==1](#) and [!ELUsingAArch32\(EL2\)](#)), this field is ignored.

If EL2 is not implemented or not enabled for the target Security state, this field is RES0.

Bits [15:9]

Reserved, RES0.

GASID, bit [8]

Execution of this instruction applies to all ASIDs or a specified ASID.

GASID	Meaning
0b0	Applies to specified ASID for an EL0 target execution context.
0b1	Applies to all ASIDs for an EL0 target execution context.

For target execution contexts other than EL0, this field is RES0.

If the instruction is executed at EL0, this field has an Effective value of 0.

ASID, bits [7:0]

Only applies for an EL0 target execution context and when bit[8] is 0.

Otherwise, this field is RES0.

When the instruction is executed at EL0, this field is treated as the current ASID.

Executing the DVPRCTX instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0111	0b0011	0b101

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTL_R_EL1.EnRCTX ==
'0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && SCTL_R_EL1.EnRCTX == '0' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T7 == '1'
then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
                AArch32.TakeHypTrapException(0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DVPRCTX ==
'1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCTL_R_EL2.EnRCTX ==
'0' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            else
                AArch32.RestrictPrediction(R[t], RestrictType_DataValue);
        elsif PSTATE.EL == EL1 then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
                AArch32.TakeHypTrapException(0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.NV == '1' then
                AArch64.SystemAccessTrap(EL2, 0x03);
            else
                AArch32.RestrictPrediction(R[t], RestrictType_DataValue);
        elsif PSTATE.EL == EL2 then
            AArch32.RestrictPrediction(R[t], RestrictType_DataValue);
        elsif PSTATE.EL == EL3 then
            AArch32.RestrictPrediction(R[t], RestrictType_DataValue);

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_AP1R<n>, Interrupt Controller Active Priorities Group 1 Registers, n = 0 - 3

The ICC_AP1R<n> characteristics are:

Purpose

Provides information about Group 1 active priorities.

Configuration

AArch32 System register ICC_AP1R<n> bits [31:0] (ICC_AP1R<n>_S) are architecturally mapped to AArch64 System register [ICC_AP1R<n>_EL1\[31:0\]](#) (ICC_AP1R<n>_EL1_S).

AArch32 System register ICC_AP1R<n> bits [31:0] (ICC_AP1R<n>_NS) are architecturally mapped to AArch64 System register [ICC_AP1R<n>_EL1\[31:0\]](#) (ICC_AP1R<n>_EL1_NS).

This register is present only when EL1 is capable of using AArch32 and FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_AP1R<n> are UNDEFINED.

Attributes

ICC_AP1R<n> is a 32-bit register.

This register has the following instances:

- ICC_AP1R<n>, when when EL3 is not implemented
- ICC_AP1R<n>_S, when when EL3 is implemented
- ICC_AP1R<n>_NS, when when EL3 is implemented

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPLEMENTATION DEFINED																															

IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Accessing ICC_AP1R<n>

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1 is only implemented in implementations that support 6 or more bits of preemption. ICC_AP1R2 and ICC_AP1R3 are only implemented in implementations that support 7 bits of preemption. Unimplemented registers are UNDEFINED.

Note

The number of bits of preemption is indicated by [ICH_VTR](#).PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- [ICC_AP0R<n>](#)
- Secure ICC_AP1R<n>
- Non-secure ICC_AP1R<n>

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-3

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(opc2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elsif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        R[t] = ICV_AP1R[m];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        R[t] = ICV_AP1R[m];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            R[t] = ICC_AP1R_NS[m];
        else
            R[t] = ICC_AP1R[m];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            UNDEFINED;
        elsif ICC_HSRE.SRE == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            R[t] = ICC_AP1R_NS[m];
        else
            R[t] = ICC_AP1R[m];
    elsif PSTATE.EL == EL3 then
        if ICC_MSRE.SRE == '0' then
            UNDEFINED;
        else

```

```

if SCR.NS == '0' then
    R[t] = ICC_AP1R_S[m];
else
    R[t] = ICC_AP1R_NS[m];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-3

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(opc2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elsif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        ICV_AP1R[m] = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        ICV_AP1R[m] = R[t];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            ICC_AP1R_NS[m] = R[t];
        else
            ICC_AP1R[m] = R[t];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            UNDEFINED;
        elsif ICC_HSRE.SRE == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            ICC_AP1R_NS[m] = R[t];
        else
            ICC_AP1R[m] = R[t];
    elsif PSTATE.EL == EL3 then
        if ICC_MSRE.SRE == '0' then
            UNDEFINED;
        else

```



```
if SCR.NS == '0' then
    ICC_AP1R_S[m] = R[t];
else
    ICC_AP1R_NS[m] = R[t];
```

3005/0907/2022 1517:5707: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_BPR1, Interrupt Controller Binary Point Register 1

The ICC_BPR1 characteristics are:

Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

Configuration

AArch32 System register ICC_BPR1 bits [31:0] (ICC_BPR1_S) are architecturally mapped to AArch64 System register [ICC_BPR1_EL1\[31:0\]](#) (ICC_BPR1_EL1_S).

AArch32 System register ICC_BPR1 bits [31:0] (ICC_BPR1_NS) are architecturally mapped to AArch64 System register [ICC_BPR1_EL1\[31:0\]](#) (ICC_BPR1_EL1_NS).

This register is present only when EL1 is capable of using AArch32 and FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_BPR1 are UNDEFINED.

In GIC implementations supporting two Security states, this register is Banked.

Attributes

ICC_BPR1 is a 32-bit register.

This register has the following instances:

- [ICC_BPR1](#), when when EL3 is not implemented
- [ICC_BPR1_S](#), when when EL3 is implemented
- [ICC_BPR1_NS](#), when when EL3 is implemented

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																	BinaryPoint														

Bits [31:3]

Reserved, RES0.

BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see 'Priority grouping' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Writing 0 to this field will set this field to its reset value.

If EL3 is implemented and [ICC_MCTLR](#).CBPR_EL1S is 1:

- Accesses to this register at EL3 not in Monitor mode access the state of [ICC_BPR0](#).
- When [SCR_EL3](#).EEL2 is 1 and [HCR_EL2](#).IMO is 1, Secure accesses to this register at EL1 access the state of [ICV_BPR1](#).
- Otherwise, Secure accesses to this register at EL1 access the state of [ICC_BPR0](#).

If EL3 is implemented and [ICC_MCTLR.CBPR_EL1NS](#) is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR.IMO and SCR.IRQ:

HCR.IMO	SCR.IRQ	Behavior
0b0	0b0	Non-secure EL1 and EL2 reads return ICC_BPR0 + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0b0	0b1	Non-secure EL1 and EL2 accesses trap to EL3.
0b1	0b0	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0 + 1 saturated to 0b111. Non-secure EL2 writes ignored.
0b1	0b1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 accesses trap to EL3.

If EL3 is not implemented and [ICC_CTLR.CBPR](#) is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR.IMO:

HCR.IMO	Behavior
0b0	Non-secure EL1 and EL2 reads return ICC_BPR0 + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0b1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0 + 1 saturated to 0b111. Non-secure EL2 writes are ignored.

This field resets to an IMPLEMENTATION DEFINED non-zero value.

Accessing ICC_BPR1

When the PE resets into an Exception level that is using AArch32, the reset value is equal to:

- For the Secure copy of the register, the minimum value of [ICC_BPR0](#) plus one.
- For the Non-secure copy of the register, the minimum value of [ICC_BPR0](#).

Where the minimum value of [ICC_BPR0](#) is IMPLEMENTATION DEFINED.

If EL3 is not implemented:

- If the PE is Secure this reset value is (minimum value of [ICC_BPR0](#) plus one).
- If the PE is Non-secure this reset value is (minimum value of [ICC_BPR0](#)).

An attempt to program the binary point field to a value less than the reset value sets the field to the reset value.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        R[t] = ICV_BPR1;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        R[t] = ICV_BPR1;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_BPR1_NS;
    else
        R[t] = ICC_BPR1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_BPR1_NS;
    else
        R[t] = ICC_BPR1;
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            R[t] = ICC_BPR1_S;
        else
            R[t] = ICC_BPR1_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        ICV_BPR1 = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        ICV_BPR1 = R[t];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_BPR1_NS = R[t];
    else
        ICC_BPR1 = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_BPR1_NS = R[t];
    else
        ICC_BPR1 = R[t];
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            ICC_BPR1_S = R[t];
        else
            ICC_BPR1_NS = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_CTLR, Interrupt Controller Control Register

The ICC_CTLR characteristics are:

Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configuration

AArch32 System register ICC_CTLR bits [31:0] (ICC_CTLR_S) are architecturally mapped to AArch64 System register [ICC_CTLR_EL1\[31:0\]](#) (ICC_CTLR_EL1_S).

AArch32 System register ICC_CTLR bits [31:0] (ICC_CTLR_NS) are architecturally mapped to AArch64 System register [ICC_CTLR_EL1\[31:0\]](#) (ICC_CTLR_EL1_NS).

This register is present only when EL1 is capable of using AArch32 and FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_CTLR are UNDEFINED.

Attributes

ICC_CTLR is a 32-bit register.

This register has the following instances:

- ICC_CTLR, when when EL3 is not implemented
- ICC_CTLR_S, when when EL3 is implemented
- ICC_CTLR_NS, when when EL3 is implemented

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												ExtRange	RSS	RES0	A3V	SEIS	IDbits	PRIbits	RES0	PMHE	RES0		EOImode	CBPR							

Bits [31:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

ExtRange	Meaning
0b0	CPU interface does not support INTIDs in the range 1024..8191. BehaviorBehaviour is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. <div>Note Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.</div>
0b1	CPU interface supports INTIDs in the range 1024..8191. All INTIDs in the range 1024..8191 are treated as requiring deactivation.

If EL3 is implemented, ICC_CTLR_EL1.ExtRange is an alias of [ICC_CTLR_EL3.ExtRange](#).

RSS, bit [18]

Range Selector Support. Possible values are:

RSS	Meaning
0b0	Targeted SGIs with affinity level 0 values of 0 - 15 are supported.
0b1	Targeted SGIs with affinity level 0 values of 0 - 255 are supported.

This bit is read-only.

Bits [17:16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0b0	The CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
0b1	The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

If EL3 is implemented and using AArch32, this bit is an alias of [ICC_MCTLR.A3V](#).

If EL3 is implemented and using AArch64, this bit is an alias of [ICC_CTLR_EL3.A3V](#).

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs:

SEIS	Meaning
0b0	The CPU interface logic does not support local generation of SEIs.
0b1	The CPU interface logic supports local generation of SEIs.

If EL3 is implemented and using AArch32, this bit is an alias of [ICC_MCTLR.SEIS](#).

If EL3 is implemented and using AArch64, this bit is an alias of [ICC_CTLR_EL3.SEIS](#).

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
0b000	16 bits.
0b001	24 bits.

All other values are reserved.

If EL3 is implemented and using AArch32, this field is an alias of [ICC_MCTLR.IDbits](#).

If EL3 is implemented and using AArch64, this field is an alias of [ICC_CTLR_EL3.IDbits](#).

PRIbits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

Note

This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of [GICD_CTLR.DS](#).

The division between group priority and subpriority is defined in the binary point registers [ICC_BPR0](#) and [ICC_BPR1](#).

If EL3 is implemented and using AArch32, physical accesses return the value from [ICC_MCTLR.PRIBits](#).

If EL3 is implemented and using AArch64, physical accesses return the value from [ICC_CTLR_EL3.PRIBits](#).

If EL3 is not implemented, physical accesses return the value from this field.

Bit [7]

Reserved, RES0.

PMHE, bit [6]

Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:

PMHE	Meaning
0b0	Disables use of ICC_PMR as a hint for interrupt distribution.
0b1	Enables use of ICC_PMR as a hint for interrupt distribution.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC_MCTLR.PMHE](#).
- If EL3 is using AArch64, this bit is an alias of [ICC_CTLR_EL3.PMHE](#).
- If [GICD_CTLR.DS](#) == 0, this bit is read-only.
- If [GICD_CTLR.DS](#) == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read/write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

Bits [5:2]

Reserved, RES0.

EOImode, bit [1]

EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode	Meaning
0b0	ICC_EOIR0 and ICC_EOIR1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR are UNPREDICTABLE.
0b1	ICC_EOIR0 and ICC_EOIR1 provide priority drop functionality only. ICC_DIR provides interrupt deactivation functionality.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC_MCTLR.EOImode_EL1](#){S, NS} where S or NS corresponds to the current Security state.
- If EL3 is using AArch64, this bit is an alias of [ICC_CTLR_EL3.EOImode_EL1](#){S, NS} where S or NS corresponds to the current Security state.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read/write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:

CBPR	Meaning
0b0	ICC_BPR0 determines the preemption group for Group 0 interrupts only. ICC_BPR1 determines the preemption group for Group 1 interrupts.
0b1	ICC_BPR0 determines the preemption group for both Group 0 and Group 1 interrupts.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC_MCTLR.CBPR_EL1](#){S,NS} where S or NS corresponds to the current Security state.
- If EL3 is using AArch64, this bit is an alias of [ICC_CTLR_EL3.CBPR_EL1](#){S,NS} where S or NS corresponds to the current Security state.
- If [GICD_CTLR.DS](#) == 0, this bit is read-only.
- If [GICD_CTLR.DS](#) == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read/write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

Accessing ICC_CTLR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> ==
'11' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TC == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TC == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        R[t] = ICV_CTLR;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> == '11'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_CTLR_NS;
    else
        R[t] = ICC_CTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_CTLR_NS;
    else
        R[t] = ICC_CTLR;
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            R[t] = ICC_CTLR_S;

```

```
else
    R[t] = ICC_CTLR_NS;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> ==
'11' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TC == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TC == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        ICV_CTLR = R[t];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> == '11'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_CTLR_NS = R[t];
    else
        ICC_CTLR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_CTLR_NS = R[t];
    else
        ICC_CTLR = R[t];
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            ICC_CTLR_S = R[t];

```

```
else
    ICC_CTLR_NS = R[t];
```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

ICC_IGRPEN1, Interrupt Controller Interrupt Group 1 Enable register

The ICC_IGRPEN1 characteristics are:

Purpose

Controls whether Group 1 interrupts are enabled for the current Security state.

Configuration

AArch32 System register ICC_IGRPEN1 bits [31:0] (ICC_IGRPEN1_S) are architecturally mapped to AArch64 System register [ICC_IGRPEN1_EL1\[31:0\]](#) (ICC_IGRPEN1_EL1_S).

AArch32 System register ICC_IGRPEN1 bits [31:0] (ICC_IGRPEN1_NS) are architecturally mapped to AArch64 System register [ICC_IGRPEN1_EL1\[31:0\]](#) (ICC_IGRPEN1_EL1_NS).

This register is present only when EL1 is capable of using AArch32 and FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_IGRPEN1 are UNDEFINED.

Attributes

ICC_IGRPEN1 is a 32-bit register.

This register has the following instances:

- ICC_IGRPEN1, when when EL3 is not implemented
- ICC_IGRPEN1_S, when when EL3 is implemented
- ICC_IGRPEN1_NS, when when EL3 is implemented

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															Enable

Bits [31:1]

Reserved, RES0.

Enable, bit [0]

Enables Group 1 interrupts for the current Security state.

Enable	Meaning
0b0	Group 1 interrupts are disabled for the current Security state.
0b1	Group 1 interrupts are enabled for the current Security state.

Virtual accesses to this register update [ICH_VMCR.VENG1](#).

If EL3 is present:

- This bit is a read/write alias of [ICC_MGRPEN1.EnableGrp1{S, NS}](#) as appropriate if EL3 is using AArch32, or [ICC_IGRPEN1_EL3.EnableGrp1{S, NS}](#) as appropriate if EL3 is using AArch64.

- When this register is accessed at EL3, the copy of this register appropriate to the current setting of SCR.NS is accessed.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_IGRPEN1

The lowest Exception level at which this register can be accessed is governed by the Exception level to which IRQ is routed. This routing depends on SCR.IRQ, SCR.NS and HCR.IMO.

If an interrupt is pending within the CPU interface when Enable becomes 0, the interrupt must be released to allow the Distributor to forward the interrupt to a different PE.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        R[t] = ICV_IGRPEN1;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        R[t] = ICV_IGRPEN1;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            R[t] = ICC_IGRPEN1_NS;
        else
            R[t] = ICC_IGRPEN1;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            UNDEFINED;
        elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            UNDEFINED;
        elsif ICC_HSRE.SRE == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch32.TakeMonitorTrapException();
        elsif HaveEL(EL3) then
            R[t] = ICC_IGRPEN1_NS;
        else
            R[t] = ICC_IGRPEN1;
    elsif PSTATE.EL == EL3 then
        if ICC_MSRE.SRE == '0' then
            UNDEFINED;
        else
            if SCR.NS == '0' then
                R[t] = ICC_IGRPEN1_S;
            else
                R[t] = ICC_IGRPEN1_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif ICC_SRE.SRE == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TALL1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        ICV_IGRPEN1 = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        ICV_IGRPEN1 = R[t];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_IGRPEN1_NS = R[t];
    else
        ICC_IGRPEN1 = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_IGRPEN1_NS = R[t];
    else
        ICC_IGRPEN1 = R[t];
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            ICC_IGRPEN1_S = R[t];
        else
            ICC_IGRPEN1_NS = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_SRE, Interrupt Controller System Register Enable register

The ICC_SRE characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL0 and EL1.

Configuration

AArch32 System register ICC_SRE bits [31:0] (ICC_SRE_S) are architecturally mapped to AArch64 System register [ICC_SRE_EL1\[31:0\]](#) (ICC_SRE_EL1_S).

AArch32 System register ICC_SRE bits [31:0] (ICC_SRE_NS) are architecturally mapped to AArch64 System register [ICC_SRE_EL1\[31:0\]](#) (ICC_SRE_EL1_NS).

This register is present only when EL1 is capable of using AArch32 and FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_SRE are UNDEFINED.

Attributes

ICC_SRE is a 32-bit register.

This register has the following instances:

- ICC_SRE, when when EL3 is not implemented
- ICC_SRE_S, when when EL3 is implemented
- ICC_SRE_NS, when when EL3 is implemented

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																DIBDFBSRE															

Bits [31:3]

Reserved, RES0.

DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0b0	IRQ bypass enabled.
0b1	IRQ bypass disabled.

If EL3 is implemented and [GICD_CTLR](#).DS == 0, this field is a read-only alias of [ICC_MSRE](#).DIB.

If EL3 is implemented and [GICD_CTLR](#).DS == 1, and EL2 is not implemented, this field is a read/write alias of [ICC_MSRE](#).DIB.

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of [ICC_HSRE](#).DIB.

If [GICD_CTLR](#).DS == 1 and EL2 is implemented, this field is a read-only alias of [ICC_HSRE](#).DIB.

In systems that do not support IRQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0b0	FIQ bypass enabled.
0b1	FIQ bypass disabled.

If EL3 is implemented and [GICD_CTLR.DS](#) == 0, this field is a read-only alias of [ICC_MSRE.DFB](#).

If EL3 is implemented and [GICD_CTLR.DS](#) == 1, and EL2 is not implemented, this field is a read/write alias of [ICC_MSRE.DFB](#).

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of [ICC_HSRE.DFB](#).

If [GICD_CTLR.DS](#) == 1 and EL2 is implemented, this field is a read-only alias of [ICC_HSRE.DFB](#).

In systems that do not support FIQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

SRE, bit [0]

System Register Enable.

SRE	Meaning
0b0	The memory-mapped interface must be used. Accesses at EL1 to any ICC_* System register other than ICC_SRE are UNDEFINED.
0b1	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and using AArch64:

- When [ICC_SRE_EL3.SRE](#)==0 the Secure copy of this bit is RAZ/WI.
- When [ICC_SRE_EL3.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

If EL3 is implemented and using AArch32:

- When [ICC_MSRE.SRE](#)==0 the Secure copy of this bit is RAZ/WI.
- When [ICC_MSRE.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

If EL2 is implemented and using AArch64:

- When [ICC_SRE_EL2.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

If EL2 is implemented and using AArch32:

- When [ICC_HSRE.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_SRE

The GIC architecture permits, but does not require, that registers can be shared between memory-mapped registers and the equivalent System registers. This means that if the memory-mapped registers have been accessed while `ICC_SRE.SRE==0`, then the System registers might be modified. Therefore, software must only rely on the reset values of the System registers if there has been no use of the GIC functionality while the memory-mapped registers are in use. Otherwise, the System register values must be treated as UNKNOWN.

Accesses to this register use the following encodings in the System register encoding space:

`MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}`

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICC_SRE_EL2.Enable == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICC_HSRE.Enable == '0' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && ICC_MSRE.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            R[t] = ICC_SRE_S;
        else
            R[t] = ICC_SRE_NS;
    else
        R[t] = ICC_SRE;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif ICC_MSRE.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            R[t] = ICC_SRE_S;
        else
            R[t] = ICC_SRE_NS;
    else
        R[t] = ICC_SRE;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        R[t] = ICC_SRE_S;
    else
        R[t] = ICC_SRE_NS;

```


MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICC_SRE_EL2.Enable == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICC_HSRE.Enable == '0' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && ICC_MSRE.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_S = R[t];
        else
            ICC_SRE_NS = R[t];
    else
        ICC_SRE = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && ICC_SRE_EL3.Enable == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif ICC_MSRE.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_S = R[t];
        else
            ICC_SRE_NS = R[t];
    else
        ICC_SRE = R[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        ICC_SRE_S = R[t];
    else
        ICC_SRE_NS = R[t];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d961b0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICV_CTLR, Interrupt Controller Virtual Control Register

The ICV_CTLR characteristics are:

Purpose

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configuration

AArch32 System register ICV_CTLR bits [31:0] are architecturally mapped to AArch64 System register [ICV_CTLR_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32, FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_CTLR are UNDEFINED.

Attributes

ICV_CTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												ExtRange	RSS	RES0	A3V	SEIS	IDbits	PRIbits	RES0								EOLmode	CBPR			

Bits [31:20]

Reserved, RES0.

ExtRange, bit [19]

Extended INTID range (read-only).

ExtRange	Meaning
0b0	CPU interface does not support INTIDs in the range 1024..8191. Behavior is UNPREDICTABLE if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface. Note Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them.
0b1	CPU interface supports INTIDs in the range 1024..8191. All INTIDs in the range 1024..8191 are treated as requiring deactivation.

ICV_CTLR.ExtRange is an alias of [ICC_CTLR](#).ExtRange.

RSS, bit [18]

Range Selector Support. Possible values are:

RSS	Meaning
0b0	Targeted SGIs with affinity level 0 values of 0 - 15 are supported.
0b1	Targeted SGIs with affinity level 0 values of 0 - 255 are supported.

This bit is read-only.

Bits [17:16]

Reserved, RES0.

A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0b0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
0b1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs:

SEIS	Meaning
0b0	The virtual CPU interface logic does not support local generation of SEIs.
0b1	The virtual CPU interface logic supports local generation of SEIs.

IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported:

IDbits	Meaning
0b000	16 bits.
0b001	24 bits.

All other values are reserved.

PRbits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation must implement at least 32 levels of physical priority (5 priority bits).

Note

This field always returns the number of priority bits implemented.

The division between group priority and subpriority is defined in the binary point registers [ICV_BPR0](#) and [ICV_BPR1](#).

Bits [7:2]

Reserved, RES0.

EOImode, bit [1]

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

EOImode	Meaning
0b0	ICV_EOIR0 and ICV_EOIR1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR are UNPREDICTABLE.
0b1	ICV_EOIR0 and ICV_EOIR1 provide priority drop functionality only. ICV_DIR provides interrupt deactivation functionality.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:

CBPR	Meaning
0b0	ICV_BPR0 determines the preemption group for virtual Group 0 interrupts only. ICV_BPR1 determines the preemption group for virtual Group 1 interrupts.
0b1	Non-secure reads of ICV_BPR1 return ICV_BPR0 plus one, saturated to 0b111. Non-secure writes to ICV_BPR1 are ignored. Secure reads of ICV_BPR1 return ICV_BPR0 . Secure writes of ICV_BPR1 modify ICV_BPR0 .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing ICV_CTLR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> ==
'11' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TC == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TC == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FMO == '1' then
        R[t] = ICV_CTLR;
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        R[t] = ICV_CTLR;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> == '11'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_CTLR_NS;
    else
        R[t] = ICC_CTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        R[t] = ICC_CTLR_NS;
    else
        R[t] = ICC_CTLR;
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            R[t] = ICC_CTLR_S;

```

```
else
    R[t] = ICC_CTLR_NS;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b1100	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> ==
'11' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && ICH_HCR_EL2.TC == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && ICH_HCR.TC == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.IMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FMO == '1' then
        ICV_CTLR = R[t];
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.IMO == '1' then
        ICV_CTLR = R[t];
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && PSTATE.M != M32_Monitor && SCR.<IRQ,FIQ> == '11'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_CTLR_NS = R[t];
    else
        ICC_CTLR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_HSRE.SRE == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) && SCR.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch32.TakeMonitorTrapException();
    elsif HaveEL(EL3) then
        ICC_CTLR_NS = R[t];
    else
        ICC_CTLR = R[t];
elsif PSTATE.EL == EL3 then
    if ICC_MSRE.SRE == '0' then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            ICC_CTLR_S = R[t];

```

```
else
    ICC_CTLR_NS = R[t];
```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

ID_DFR0, Debug Feature Register 0

The ID_DFR0 characteristics are:

Purpose

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch32 System register ID_DFR0 bits [31:0] are architecturally mapped to AArch64 System register [ID_DFR0_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to ID_DFR0 are UNDEFINED.

Attributes

ID_DFR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TraceFilt				PerfMon				MProfDbg				MMapTrc				CopTrc				MMapDbg				CopSDBG				CopDBG			

TraceFilt, bits [31:28]

Armv8.4 Self-hosted Trace Extension version. Defined values are:

TraceFilt	Meaning
0b0000	Armv8.4 Self-hosted Trace Extension not implemented.
0b0001	Armv8.4 Self-hosted Trace Extension implemented.

All other values are reserved.

FEAT_TRF implements the functionality added by the value 0b0001.

From Armv8.3, the permitted values are 0b0000 and 0b0001.

PerfMon, bits [27:24]

Performance Monitors Extension version.

This field does not follow the standard ID scheme, but uses the alternative ID scheme described in 'Alternative ID scheme used for the Performance Monitors Extension version'.

Defined values are:

PerfMon	Meaning
0b0000	Performance Monitors Extension not implemented.
0b0001	Performance Monitors Extension, PMUv1 implemented.
0b0010	Performance Monitors Extension, PMUv2 implemented.
0b0011	Performance Monitors Extension, PMUv3 implemented.
0b0100	PMUv3 for Armv8.1. As 0b0011, and adds support for: <ul style="list-style-type: none"> Extended 16-bit PMEVTYPER<n>.evtCount field. If EL2 is implemented, the HDCR.HPMD control.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds support for the PMMIR register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the HDCR.HCCD control. If EL3 is implemented, the SDCR.SCCD control.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds support for: <ul style="list-style-type: none"> The PMCR.FZO and, if EL2 is implemented, HDCR.HPMFZO controls. If EL3 is implemented and using AArch64, the MDCR_EL3.{MPMX,MCCD} controls.
0b1000	PMUv3 for Armv8.8. As 0b0111, and: <ul style="list-style-type: none"> Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. Removes the CONSTRAINED UNPREDICTABLE behaviors if a reserved or unimplemented PMU event number is selected.
0b1001	PMUv3 for Armv8.9. As 0b1000, and: <ul style="list-style-type: none"> Updates the definitions of existing PMU events. Adds support for the EDECR.PME control.
0b1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. Arm does not recommend this value for new implementations.

All other values are reserved.

FEAT_PMUv3 implements the functionality identified by the value 0b0011.

FEAT_PMUv3p1 implements the functionality identified by the value 0b0100.

FEAT_PMUv3p4 implements the functionality identified by the value 0b0101.

FEAT_PMUv3p5 implements the functionality identified by the value 0b0110.

FEAT_PMUv3p7 implements the functionality identified by the value 0b0111.

FEAT_PMUv3p8 implements the functionality identified by the value 0b1000.

FEAT_PMUv3p9 implements the functionality identified by the value 0b1001.

In any Armv8 implementation, the values 0b0001 and 0b0010 are not permitted.

From Armv8.1, if FEAT_PMUv3 is implemented, the value 0b0011 is not permitted.

From Armv8.4, if FEAT_PMUv3 is implemented, the value 0b0100 is not permitted.

From Armv8.5, if FEAT_PMUv3 is implemented, the value 0b0101 is not permitted.

From Armv8.7, if FEAT_PMUv3 is implemented, the value 0b0110 is not permitted.

From Armv8.8, if FEAT_PMUv3 is implemented, the value 0b0111 is not permitted.

Note

PMUv1 and PMUv2 are not permitted in an Armv8 implementation.

From Armv8.9, if FEAT_PMUv3 is implemented, the value 0b1000 is not permitted.

Note

In Armv7, the value 0b0000 can mean that PMUv1 is implemented. PMUv1 and PMUv2 are not permitted in an Armv8 implementation.

MProfDbg, bits [23:20]

M-profile Debug. Support for memory-mapped debug model for M-profile processors. Defined values are:

MProfDbg	Meaning
0b0000	Not supported.
0b0001	Support for M-profile Debug architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

MMapTrc, bits [19:16]

Memory-mapped Trace. Support for memory-mapped trace model. Defined values are:

MMapTrc	Meaning
0b0000	Not supported.
0b0001	Support for Arm trace architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the permitted values are 0b0000 and 0b0001.

For more information, see the ARM® Embedded Trace Macrocell Architecture Specification, ETMv4 (ARM IHI 0064).

CopTrc, bits [15:12]

Support for System registers-based trace model, using registers in the coproc == 0b1110 encoding space. Defined values are:

CopTrc	Meaning
0b0000	Not supported.
0b0001	Support for Arm trace architecture, with System registers access.

All other values are reserved.

In Armv8-A, the permitted values are 0b0000 and 0b0001.

For more information, see the ARM® Embedded Trace Macrocell Architecture Specification, ETMv4 (ARM IHI 0064).

MMapDbg, bits [11:8]

Memory-mapped Debug. Support for Armv7 memory-mapped debug model for A and R-profile processors. Defined values are:

MMapDbg	Meaning
0b0000	Not supported.
0b0100	Support for Armv7, v7 Debug architecture, with memory-mapped access.
0b0101	Support for Armv7, v7.1 Debug architecture, with memory-mapped access.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

The optional memory map defined by Armv8 is not compatible with Armv7.

CopSDBG, bits [7:4]

Support for a System registers-based Secure debug model, using registers in the coproc = 0b1110 encoding space, for an A-profile processor that includes EL3.

If EL3 is not implemented and the implemented Security state is Non-secure state, this field is RES0. Otherwise, this field reads the same as bits [3:0].

CopDBG, bits [3:0]

Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are:

CopDBG	Meaning
0b0000	Not supported.
0b0010	Armv6, v6 Debug architecture, with System registers access.
0b0011	Armv6, v6.1 Debug architecture, with System registers access.
0b0100	Armv7, v7 Debug architecture, with System registers access.
0b0101	Armv7, v7.1 Debug architecture, with System registers access.
0b0110	Armv8 debug architecture.
0b0111	Armv8 debug architecture with Virtualization Host Extensions.
0b1000	Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.
0b1011	Armv8.9 debug architecture, FEAT_Debugv8p9.

All other values are reserved.

The values 0b0000, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8.

FEAT_VHE adds the functionality identified by the value 0b0111.

FEAT_Debugv8p2 adds the functionality identified by the value 0b1000.

FEAT_Debugv8p4 adds the functionality identified by the value 0b1001.

FEAT_Debugv8p8 adds the functionality identified by the value 0b1010.

FEAT_Debugv8p9 adds the functionality identified by the value 0b1011.

From Armv8.1, when FEAT_VHE is implemented the value 0b0110 is not permitted.

From Armv8.2, the values 0b0110 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 is not permitted.

From Armv8.9, the value 0b1010 is not permitted.

Accessing ID_DFR0

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0000	0b0001	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TID3 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TID3 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        R[t] = ID_DFR0;
elsif PSTATE.EL == EL2 then
    R[t] = ID_DFR0;
elsif PSTATE.EL == EL3 then
    R[t] = ID_DFR0;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_ISAR6, Instruction Set Attribute Register 6

The ID_ISAR6 characteristics are:

Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID_ISAR0](#), [ID_ISAR1](#), [ID_ISAR2](#), [ID_ISAR3](#), [ID_ISAR4](#), and [ID_ISAR5](#).

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch32 System register ID_ISAR6 bits [31:0] are architecturally mapped to AArch64 System register [ID_ISAR6_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to ID_ISAR6 are UNDEFINED.

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_ISAR6 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRBHB	RES0			I8MM					BF16			SPECRES			SB				FHM				DP				JSCVT				

CLRBHB, bits [31:28]

Indicates support for the CLRBHB instruction in AArch32 state. Defined values are:

Reserved, RES0.

CLRBHB	Meaning
0b0000	CLRBHB instruction is not implemented.
0b0001	CLRBHB instruction is implemented.

All other values are reserved.

FEAT_CLRBHB implements the functionality identified by 0b0001.

From Armv8.9, the only permitted value is 0b0001.

I8MM, bits [27:24]

Indicates support for Advanced SIMD and floating-point Int8 matrix multiplication instructions in AArch32 state. Defined values are:

I8MM	Meaning
0b0000	Int8 matrix multiplication instructions are not implemented.
0b0001	VSMMLA, VSUDOT, VUMMLA, VUSMMLA, and VUSDOT instructions are implemented.

All other values are reserved.

FEAT_AA32I8MM implements the functionality identified by 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

BF16, bits [23:20]

Indicates support for Advanced SIMD and floating-point BFloat16 instructions in AArch32 state. Defined values are:

BF16	Meaning
0b0000	BFloat16 instructions are not implemented.
0b0001	VCVT, VCVTB, VCVTT, VDOT, VFMA, VFMA, and VMMLA instructions with BF16 operand or result types are implemented.

All other values are reserved.

FEAT_AA32BF16 implements the functionality identified by 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

SPECRES, bits [19:16]

Indicates support for **predictionSpeculation** invalidation instructions in AArch32 state. Defined values are:

SPECRES	Meaning
0b0000	PredictionCFPRCTX, invalidationDVPRCTX, and CPPRCTX instructions are not implemented.
0b0001	CFPRCTX, DVPRCTX, and CPPRCTX instructions are implemented.
0b0010	As 0b0001, and the COSPRCTX instruction is implemented.

All other values are reserved.

FEAT_SPECRES implements **From Armv8.5, the functionality only identifiedpermitted byvalue is** 0b0001.

FEAT_SPECRES2 implements the functionality identified by **0b0010**.

From Armv8.5, the value 0b0000 is not permitted.

From Armv8.9, the value 0b0001 is not permitted.

SB, bits [15:12]

Indicates support for SB instruction in AArch32 state. Defined values are:

SB	Meaning
0b0000	SB instruction is not implemented.
0b0001	SB instruction is implemented.

All other values are reserved.

From Armv8.5, the only permitted value is 0b0001.

FHM, bits [11:8]

Indicates support for Advanced SIMD and floating-point VFMA and VFMSL instructions in AArch32 state. Defined values are:

FHM	Meaning
0b0000	VFMAI and VMFSL instructions not implemented.
0b0001	VFMAI and VMFSL instructions implemented.

FEAT_FHM implements the functionality identified by the value 0b0001.

DP, bits [7:4]

Indicates support for dot product instructions in AArch32 state. Defined values are:

DP	Meaning
0b0000	No dot product instructions implemented.
0b0001	VUDOT and VSDOT instructions implemented.

All other values are reserved.

FEAT_DotProd implements the functionality identified by the value 0b0001.

JSCVT, bits [3:0]

Indicates support for the Javascript conversion instruction in AArch32 state. Defined values are:

JSCVT	Meaning
0b0000	The VJCVT instruction is not implemented.
0b0001	The VJCVT instruction is implemented.

All other values are reserved.

In Armv8.0, the only permitted value is 0b0000.

FEAT_JSCVT implements the functionality identified by 0b0001.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is 0b0001.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is 0b0000.

Accessing ID_ISAR6

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0000	0b0010	0b111


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && (IsFeatureImplemented(FEAT_FGT) ||
!IsZero(ID_ISAR6) || boolean IMPLEMENTATION_DEFINED "ID_ISAR6 trapped by HCR_EL2.TID3") &&
HCR_EL2.TID3 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && (IsFeatureImplemented(FEAT_FGT) ||
!IsZero(ID_ISAR6) || boolean IMPLEMENTATION_DEFINED "ID_ISAR6 trapped by HCR.TID3") && HCR.TID3 ==
'1' then
        AArch32.TakeHypTrapException(0x03);
    else
        R[t] = ID_ISAR6;
elsif PSTATE.EL == EL2 then
    R[t] = ID_ISAR6;
elsif PSTATE.EL == EL3 then
    R[t] = ID_ISAR6;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_PFR0, Processor Feature Register 0

The ID_PFR0 characteristics are:

Purpose

Gives top-level information about the instruction sets and other features supported by the PE in AArch32 state.

Must be interpreted with [ID_PFR1](#).

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch32 System register ID_PFR0 bits [31:0] are architecturally mapped to AArch64 System register [ID_PFR0_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to ID_PFR0 are UNDEFINED.

Attributes

ID_PFR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAS				DIT				AMU				CSV2				State3				State2				State1				State0			

RAS, bits [31:28]

RAS Extension version. ~~Defined values are:~~

RAS	Meaning
0b0000	No RAS Extension.
0b0001	RAS Extension implemented.
0b0010	FEAT_RASv1p1 implemented. As 0b0001, and adds support for additional ERXMISC<m> System registers. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ERR<n>STATUS and support for the optional RAS Timestamp Extension.
0b0011	FEAT_RASv2 implemented. As 0b0010, and requires that error records accessed through System registers conform to RAS System Architecture v2.

All other values are reserved.

FEAT_RAS implements the functionality identified by the value 0b0001.

FEAT_RASv1p1 implements the functionality identified by the value 0b0010.

FEAT_RASv2 implements the functionality identified by the value 0b0011.

In Armv8.0 and Armv8.1, the permitted values are 0b0000 and 0b0001.

From Armv8.4, when FEAT_DoubleFault is not implemented, and [ERRIDR.NUM](#) is 0, the permitted values are IMPLEMENTATION DEFINED 0b0001 or 0b0010.

From In Armv8.2, the only permitted value is 0b00000b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented or [ERRIDR.NUM](#) is nonzero, the value 0b0001 is not permitted.

From Armv8.4, if FEAT_DoubleFault is implemented, the only permitted value is 0b0010.

Note

When the value of this field is 0b0001, [ID_PFR2.RAS_frac](#) indicates whether FEAT_RASv1p1 is implemented.

DIT, bits [27:24]

Data Independent Timing. Defined values are:

DIT	Meaning
0b0000	AArch32 does not guarantee constant execution time of any instructions.
0b0001	AArch32 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.

All other values are reserved.

FEAT_DIT implements the functionality identified by the value 0b0001.

From Armv8.4, the only permitted value is 0b0001.

AMU, bits [23:20]

Indicates support for Activity Monitors Extension. Defined values are:

AMU	Meaning
0b0000	Activity Monitors Extension is not implemented.
0b0001	FEAT_AMUv1 is implemented.
0b0010	FEAT_AMUv1p1 is implemented. As 0b0001 and adds support for virtualization of the activity monitor event counters.

All other values are reserved.

FEAT_AMUv1 implements the functionality identified by the value 0b0001.

FEAT_AMUv1p1 implements the functionality identified by the value 0b0010.

In Armv8.0, the only permitted value is 0b0000.

In Armv8.4, the permitted values are 0b0000 and 0b0001.

From Armv8.6, the permitted values are 0b0000, 0b0001, and 0b0010.

CSV2, bits [19:16]

Speculative use of out of context branch targets. Defined values are:

CSV2	Meaning
0b0000	The implementation does not disclose whether FEAT_CSV2 is implemented.
0b0001	FEAT_CSV2 is implemented, but FEAT_CSV2_1p1 is not implemented.
0b0010	FEAT_CSV2_1p1 is implemented.

All other values are reserved.

FEAT_CSV2 implements the functionality identified by the value 0b0001.

FEAT_CSV2_1p1 implements the functionality identified by the value 0b0010.

From Armv8.5, the permitted values are 0b0001 and 0b0010.

State3, bits [15:12]

T32EE instruction set support. Defined values are:

State3	Meaning
0b0000	Not implemented.
0b0001	T32EE instruction set implemented.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

State2, bits [11:8]

Jazelle extension support. Defined values are:

State2	Meaning
0b0000	Not implemented.
0b0001	Jazelle extension implemented, without clearing of JOSCR.CV on exception entry.
0b0010	Jazelle extension implemented, with clearing of JOSCR.CV on exception entry.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0001.

State1, bits [7:4]

T32 instruction set support. Defined values are:

State1	Meaning
0b0000	T32 instruction set not implemented.
0b0001	T32 encodings before the introduction of Thumb-2 technology implemented: <ul style="list-style-type: none"> • All instructions are 16-bit. • A BL or BLX is a pair of 16-bit instructions. • 32-bit instructions other than BL and BLX cannot be encoded.
0b0011	T32 encodings after the introduction of Thumb-2 technology implemented, for all 16-bit and 32-bit T32 basic instructions.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0011.

State0, bits [3:0]

A32 instruction set support. Defined values are:

State0	Meaning
0b0000	A32 instruction set not implemented.
0b0001	A32 instruction set implemented.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0001.

(old)

htmldiff from-

(new)

PMCCFILTR, Performance Monitors Cycle Count Filter Register

The PMCCFILTR characteristics are:

Purpose

Determines the modes in which the Cycle Counter, [PMCCNTR](#), increments.

Configuration

AArch32 System register PMCCFILTR bits [31:0] are architecturally mapped to AArch64 System register [PMCCFILTR_EL0\[31:0\]](#).

AArch32 System register PMCCFILTR bits [31:0] are architecturally mapped to External register [PMU.PMCCFILTR_EL0\[31:0\]](#)[PMCCFILTR_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMCCFILTR are UNDEFINED.

Attributes

PMCCFILTR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	U	NSK	NSU	NSH	RES0				RLU	RES0																					

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR.NSK bit.

P	Meaning
0b0	Count cycles in EL1.
0b1	Do not count cycles in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR.RLU bit.

U	Meaning
0b0	Count cycles in EL0.
0b1	Do not count cycles in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of PMCCFILTR.P, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of PMCCFILTR.U, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hyp mode) filtering bit. Controls counting in EL2.

NSH	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [26:22]

Reserved, RES0.

RLU, bit [21]
When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [20:0]

Reserved, RES0.

Accessing PMCCFILTR

PMCCFILTR can also be accessed by using [PMXEVTYPER](#) with [PMSELR](#).SEL set to 0b11111.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b1111	0b111


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCFILTR;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCFILTR;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCFILTR;
elseif PSTATE.EL == EL3 then
    R[t] = PMCCFILTR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b1111	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMCCFILTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCFILTR = R[t];
elseif PSTATE.EL == EL3 then
    PMCCFILTR = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCCNTR, Performance Monitors Cycle Count Register

The PMCCNTR characteristics are:

Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' for more information.

[PMCCFILTER](#) determines the modes and states in which the PMCCNTR can increment.

Configuration

AArch32 System register PMCCNTR bits [63:0] are architecturally mapped to AArch64 System register [PMCCNTR_EL0\[63:0\]](#).

AArch32 System register PMCCNTR bits [63:0] are architecturally mapped to External register [PMU.PMCCNTR_EL0\[63:0\]](#)[PMCCNTR_EL0\[63:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMCCNTR are UNDEFINED.

PMCCNTR is a 64-bit register that can also be accessed as a 32-bit value. If it is accessed as a 32-bit register, accesses read and write bits [31:0] and do not modify bits [63:32].

All counters are subject to any changes in clock frequency, including clock stopping caused by the WFI and WFE instructions. This means that it is CONSTRAINED UNPREDICTABLE whether or not PMCCNTR continues to increment when clocks are stopped by WFI and WFE instructions.

Attributes

PMCCNTR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CCNT, bits [63:0]

Cycle count. Depending on the values of [PMCR](#).{LC,D}, this field increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR](#).C sets this field to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCCNTR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<CR,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<CR,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCCNTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCNTR<31:0>;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCNTR<31:0>;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCCNTR<31:0>;
elsif PSTATE.EL == EL3 then
    R[t] = PMCCNTR<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMCCNTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCNTR = ZeroExtend(R[t]);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCNTR = ZeroExtend(R[t]);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCCNTR = ZeroExtend(R[t]);
elsif PSTATE.EL == EL3 then
    PMCCNTR = ZeroExtend(R[t]);

```

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1001	0b0000


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<CR,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<CR,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCCNTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (PMCCNTR<63:32>, PMCCNTR<31:0>);
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (PMCCNTR<63:32>, PMCCNTR<31:0>);
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        (R[t2], R[t]) = (PMCCNTR<63:32>, PMCCNTR<31:0>);
elsif PSTATE.EL == EL3 then
    (R[t2], R[t]) = (PMCCNTR<63:32>, PMCCNTR<31:0>);

```

MCCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1001	0b0000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMCCNTR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        PMCCNTR = R[t2]:R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        PMCCNTR = R[t2]:R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x04);
    else
        PMCCNTR = R[t2]:R[t];
elsif PSTATE.EL == EL3 then
    PMCCNTR = R[t2]:R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F.

For more information about the Common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configuration

AArch32 System register PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID0_EL0\[31:0\]](#).

AArch32 System register PMCEID0 bits [31:0] are architecturally mapped to External register [PMU.PMCEID0\[31:0\]](#)[PMCEID0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCEID0 are UNDEFINED.

Attributes

PMCEID0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event n.

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID0

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b110

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID0;
elsif PSTATE.EL == EL3 then
    R[t] = PMCEID0;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x0020 to 0x003F.

For more information about the Common events and the use of the PMCEIDn registers see 'The PMU event number space and common events'.

Configuration

AArch32 System register PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID1_EL0\[31:0\]](#).

AArch32 System register PMCEID1 bits [31:0] are architecturally mapped to External register [PMU.PMCEID1\[31:0\]](#)[PMCEID1\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCEID1 are UNDEFINED.

Attributes

PMCEID1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID1

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b111

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID1;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID1;
elsif PSTATE.EL == EL3 then
    R[t] = PMCEID1;

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID2

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID2;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID2;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID2;
elsif PSTATE.EL == EL3 then
    R[t] = PMCEID2;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID3

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID3;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID3;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCEID3;
elsif PSTATE.EL == EL3 then
    R[t] = PMCEID3;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCNTENCLR, Performance Monitors Count Enable Clear register

The PMCNTENCLR characteristics are:

Purpose

Disables the Cycle Count Register, [PMCCNTR](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

PMCNTENCLR is used in conjunction with the [PMCNTENSET](#) register.

Configuration

AArch32 System register PMCNTENCLR bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENCLR_EL0\[31:0\]](#).

AArch32 System register PMCNTENCLR bits [31:0] are architecturally mapped to External register [PMU.PMCNTENCLR_EL0\[31:0\]](#)[PMCNTENCLR_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMCNTENCLR are UNDEFINED.

Attributes

PMCNTENCLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

[PMCCNTR](#) disable bit. Disables the cycle counter register.

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, disables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter disable bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n> is disabled. When written, has no effect.
0b1	When read, means that PMEVCNTR<n> is enabled. When written, disables PMEVCNTR<n> .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENCLR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b010

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCNTEN ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENCLR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENCLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENCLR;
elsif PSTATE.EL == EL3 then
    R[t] = PMCNTENCLR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b010


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCNTEN ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENCLR = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENCLR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENCLR = R[t];
elsif PSTATE.EL == EL3 then
    PMCNTENCLR = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCNTENSET, Performance Monitors Count Enable Set register

The PMCNTENSET characteristics are:

Purpose

Enables the Cycle Count Register, [PMCCNTR](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

PMCNTENSET is used in conjunction with the [PMCNTENCLR](#) register.

Configuration

AArch32 System register PMCNTENSET bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENSET_ELO\[31:0\]](#).

AArch32 System register PMCNTENSET bits [31:0] are architecturally mapped to External register [PMU.PMCNTENSET_ELO\[31:0\]](#)[PMCNTENSET_ELO\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMCNTENSET are UNDEFINED.

Attributes

PMCNTENSET is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

[PMCCNTR](#) enable bit. Enables the cycle counter register.

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, enables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter enable bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2.HPMN](#) if EL2 is using AArch64, or in [HDCR.HPMN](#) if EL2 is using AArch32. Otherwise, N is the value in [PMCR.N](#).

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n> is disabled. When written, has no effect.
0b1	When read, means that PMEVCNTR<n> event counter is enabled. When written, enables PMEVCNTR<n> .

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENSET

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMCNTEN ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENSET;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENSET;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCNTENSET;
elsif PSTATE.EL == EL3 then
    R[t] = PMCNTENSET;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b001

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCNTEN ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENSET = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENSET = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCNTENSET = R[t];
elsif PSTATE.EL == EL3 then
    PMCNTENSET = R[t];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMCR, Performance Monitors Control Register

The PMCR characteristics are:

Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configuration

AArch32 System register PMCR bits [31:0] are architecturally mapped to AArch64 System register [PMCR_EL0\[31:0\]](#).

AArch32 System register PMCR bits [7:0] are architecturally mapped to External register [PMU.PMCR_EL0\[7:0\]](#)~~[PMCR_EL0\[7:0\]](#)~~.

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCR are UNDEFINED.

Attributes

PMCR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMP								IDCODE								N		RES0	FZ	RES0	LP	LC	DP	X	D	C	P	E			

IMP, bits [31:24]

When FEAT_PMUv3p7 is not implemented:

Implementer code.

If this field is zero, then PMCR.IDCODE is RES0 and software must use [MIDR](#) to identify the PE.

Otherwise, this field and PMCR.IDCODE identify the PMU implementation to software. The implementer codes are allocated by Arm. A non-zero value has the same interpretation as [MIDR](#).Implementer.

Use of this field is deprecated.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RAZ.

IDCODE, bits [23:16]

When PMCR.IMP != 0b00000000:

Identification code. Use of this field is deprecated.

Each implementer must maintain a list of identification codes that are specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

N, bits [15:11]

Indicates the number of event counters implemented. This value is in the range of 0b000000-0b11111. If the value is 0b000000, then only [PMCCNTR](#) is implemented. If the value is 0b11111, then [PMCCNTR](#) and 31 event counters are implemented.

In an implementation that includes EL2:

- If EL2 is using AArch32, reads of this field from Non-secure EL1 and Non-secure EL0 return the value of [HDCR](#).HPMN.
- If EL2 is using AArch64 and is enabled in the current Security state, reads of this field from EL1 and EL0 return the value of [MDCR_EL2](#).HPMN.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bit [10]

Reserved, RES0.

FZO, bit [9]

When FEAT_PMUv3p7 is implemented:

Freeze-on-overflow. Stop event counters on overflow.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR.N.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counter PMEVCNTR<n> does not count when PMOVSRR [(PMN-1):0] is nonzero and n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [8]

Reserved, RES0.

LP, bit [7]**When FEAT_PMUv3p5 is implemented:**

Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR.N.

LP	Meaning
0b0	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n> [31:0].
0b1	Event counter overflow on increment that causes unsigned overflow of PMEVCNTR<n> [63:0].

If the highest implemented Exception level is using AArch32, it is IMPLEMENTATION DEFINED whether this bit is RW or RAZ/WI.

If PMN is not 0, this bit affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR](#).

[PMEVCNTR<n>](#)[63:32] cannot be accessed directly in AArch32 state.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

LC, bit [6]

Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit.

LC	Meaning
0b0	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR [31:0].
0b1	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR [63:0].

Arm deprecates use of [PMCR](#).LC = 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

DP, bit [5]**When EL3 is implemented or (FEAT_PMUv3p1 is implemented and EL2 is implemented):**

Disable cycle counter when event counting is prohibited.

DP	Meaning
0b0	Cycle counting by PMCCNTR is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR is disabled in prohibited regions and when event counting is frozen: <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and HDCR.HPMD is 1, then cycle counting by PMCCNTR is disabled at EL2. If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMCCNTR is disabled at EL3. If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR.FZO, then cycle counting by PMCCNTR is disabled. If EL3 is implemented, MDCR_EL3.SPME or SDCR.SPME is 0, and either FEAT_PMUv3p7 is not implemented, EL3 is using AArch32, or MDCR_EL3.MPMX is 0, then cycle counting by PMCCNTR is disabled at EL3 and in Secure state. <p>If HDCR.HPMN is not 0, this is when event counting by event counters in the range [0..(HDCR.HPMN-1)] is prohibited or frozen.</p>

For more information, see 'Prohibiting event and cycle counting'.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

X, bit [4]**When the implementation includes a PMU event export bus:**

Enable export of events in an IMPLEMENTATION DEFINED PMU event export bus.

X	Meaning
0b0	Do not export events.
0b1	Export events where not prohibited.

This field enables the exporting of events over an IMPLEMENTATION DEFINED PMU event export bus to another device, for example to an OPTIONAL trace unit.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RAZ/WI.

D, bit [3]

Clock divider.

D	Meaning
0b0	When enabled, PMCCNTR counts every clock cycle.
0b1	When enabled, PMCCNTR counts once every 64 clock cycles.

If PMCR.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

Arm deprecates use of PMCR.D = 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

C, bit [2]

Cycle counter reset. The effects of writing to this bit are:

C	Meaning
0b0	No action.
0b1	Reset PMCCNTR to zero.

Note

Resetting [PMCCNTR](#) does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR.LC is ignored, and bits [63:0] of the cycle counter are reset.

Access to this field is **WO/RAZ**.

P, bit [1]

Event counter reset.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR.N.

P	Meaning
0b0	No action.
0b1	If n is in the range of affected event counters, resets each event counter PMEVCNTR<n> to zero.

The effects of writing to this bit are:

- If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)].
- If EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters.
- In EL2 and EL3, a write of 1 to this bit resets all the event counters.
- This field does not affect the operation of other event counters and [PMCCNTR](#).

Note

Resetting the event counters does not change the event counter overflow bits.

If FEAT_PMUv3p5 is implemented, the values of [HDCR](#).HLP and PMCR.LP are ignored and bits [63:0] of all affected event counters are reset.

Access to this field is **WO/RAZ**.

E, bit [0]

Enable.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR.N.

E	Meaning
0b0	PMCCNTR is disabled and event counters PMEVCNTR<n> , where n is in the range of affected event counters, are disabled.
0b1	PMCCNTR and event counters PMEVCNTR<n> where n is in the range of affected event counters, are enabled by PMCNTENSET .

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMCR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b000

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPMCR == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPMCR == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPMCR == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPMCR == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMCR;

```

```
elsif PSTATE.EL == EL3 then  
    R[t] = PMCR;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b000


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCR_EL0
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPMCR == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPMCR == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCR = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPMCR == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPMCR == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;

```

```

    else
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCR = R[t];
elseif PSTATE.EL == EL3 then
    PMCR = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMEVCNTR<n>, Performance Monitors Event Count Registers, n = 0 - 30

The PMEVCNTR<n> characteristics are:

Purpose

Holds event counter n, which counts events, where n is 0 to 30.

Configuration

AArch32 System register PMEVCNTR<n> bits [31:0] are architecturally mapped to AArch64 System register [PMEVCNTR<n>_EL0\[31:0\]](#).

AArch32 System register PMEVCNTR<n> bits [31:0] are architecturally mapped to External register [PMU.PMEVCNTR<n>_EL0\[31:0\]](#)~~PMEVCNTR<n>_EL0[31:0]~~.

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMEVCNTR<n> are UNDEFINED.

Attributes

PMEVCNTR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Event counter n																															

Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

If FEAT_PMuV3p5 is implemented, the event counter is 64 bits and only the least-significant part of the event counter is accessible in AArch32 state:

- Reads from PMEVCNTR<n> return bits [31:0] of the counter.
- Writes to PMEVCNTR<n> update bits [31:0] and leave bits [63:32] unchanged.
- There is no means to access bits [63:32] directly from AArch32 state.
- If the implementation does not support AArch64, bits [63:32] are not required to be implemented.

If FEAT_PMuV3p5 is not implemented, the event counter is 32 bits.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVCNTR<n>

PMEVCNTR<n> can also be accessed by using [PMXEVCNTR](#) with [PMSELR](#).SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMEVCNTR<n>](#) is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of [PMEVCNTR<n>](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR](#).{ER,EN} or [PMUSERENR_EL0](#).{ER,EN}.

If EL2 is implemented and enabled in the current Security state, at EL0 and EL1:

- If EL2 is using AArch32, [HDCR](#).HPMN identifies the number of accessible event counters.
- If EL2 is using AArch64, [MDCR_EL2](#).HPMN identifies the number of accessible event counters.

Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [HDCR](#).HPMN and [MDCR_EL2](#).HPMN.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-30

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b10:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:opc2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && PMUSERENR.<ER,EN> == '00' then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
                AArch32.TakeHypTrapException(0x00);
            else
                UNDEFINED;
            elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
                AArch32.TakeHypTrapException(0x03);
            elsif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
                if !IsFeatureImplemented(FEAT_FGT) then
                    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
                elsif ELUsingAArch32(EL1) then
                    AArch32.TakeHypTrapException(0x03);
                else
                    AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
                if Halted() && EDSCR.SDD == '1' then
                    UNDEFINED;
                else
                    AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                R[t] = PMEVCNTR[m];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVCNTR[m];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then

```

```

    UNDEFINED;
elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVCNTR[m];
elseif PSTATE.EL == EL3 then
    R[t] = PMEVCNTR[m];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-30

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b10:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:opc2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elseif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[m] = R[t];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elseif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[m] = R[t];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then

```

```
    UNDEFINED;
elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[m] = R[t];
elseif PSTATE.EL == EL3 then
    PMEVCNTR[m] = R[t];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMEVTYPEPER<n>, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n> characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Configuration

AArch32 System register PMEVTYPER<n> bits [31:0] are architecturally mapped to AArch64 System register [PMEVTYPER<n>_EL0\[31:0\]](#).

AArch32 System register PMEVTYPER<n> bits [31:0] are architecturally mapped to External register [PMU.PMEVTYPER<n>_EL0\[31:0\]](#)~~[PMEVTYPER<n>_EL0\[31:0\]](#)~~.

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVTYPER<n> are UNDEFINED.

Attributes

PMEVTYPER<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	U	NSK	NSU	NSH	RES0	MT	RES0	RLU	RES0	evtCount[15:10]	evtCount[9:0]																				

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>.NSK bit.

P	Meaning
0b0	Count events in EL1.
0b1	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>.RLU bit.

U	Meaning
0b0	Count events in EL0.
0b1	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]
When EL3 is implemented:

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of PMEVTYPER<n>.P, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]
When EL3 is implemented:

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of PMEVTYPER<n>.U, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]
When EL2 is implemented:

EL2 (Hyp mode) filtering bit. Controls counting in EL2.

NSH	Meaning
0b0	Do not count events in EL2.
0b1	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [26]

Reserved, RES0.

MT, bit [25]

When FEAT_MTPMU is implemented or an IMPLEMENTATION DEFINED multi-threaded PMU extension is implemented:

Multithreading.

MT	Meaning
0b0	Count events only on controlling PE.
0b1	Count events from any PE with the same affinity at level 1 and above as this PE.

From Armv8.6, the IMPLEMENTATION DEFINED multi-threaded PMU extension is not permitted, meaning if FEAT_MTPMU is not implemented, this bit is RES0. See [ID_DFR1](#).MTPMU.

This bit is ignored by the PE and treated as zero when FEAT_MTPMU is implemented and Disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [24:22]

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [20:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]

When FEAT_PMUv3p1 is implemented:

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

evtCount[9:0], bits [9:0]

Event to count.

The event number of the event that is counted by event counter [PMEVCNTR<n>](#).

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.

Note

Arm recommends this behavior for all implementations of FEAT_PMUv3.

Otherwise, if PMEVTYPER<n>.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>.evtCount field is UNKNOWN.

Note

UNPREDICTABLE means the event must not expose privileged information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVTYPER<n>

PMEVTYPER<n> can also be accessed by using [PMXEVTYPER](#) with [PMSCLR](#).SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMEVTYPER<n>](#) is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of [PMEVTYPER<n>](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR](#).EN or [PMUSERENR_EL0](#).EN.

If EL2 is implemented and enabled in the current Security state, at EL0 and EL1:

- If EL2 is using AArch32, [HDCR](#).HPMN identifies the number of accessible event counters.
- If EL2 is using AArch64, [MDCR_EL2](#).HPMN identifies the number of accessible event counters.

Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [HDCR](#).HPMN and [MDCR_EL2](#).HPMN.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-30

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:opc2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMEVTYPEPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elseif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVTYPER[m];
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elseif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVTYPER[m];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then

```

```
    UNDEFINED;
elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVTYPER[m];
elseif PSTATE.EL == EL3 then
    R[t] = PMEVTYPER[m];
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>} ; Where m = 0-30

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:m[4:3]	m[2:0]

```

integer m = UInt(CRm<1:0>:opc2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMEVTYPEPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[m] = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && m >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[m] = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then

```



```
    UNDEFINED;
elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[m] = R[t];
elseif PSTATE.EL == EL3 then
    PMEVTYPER[m] = R[t];
```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMINTENCLR, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR characteristics are:

Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR](#), and the event counters [PMEVCNTR<n>](#). Reading the register shows which overflow interrupt requests are enabled.

PMINTENCLR is used in conjunction with the [PMINTENSET](#) register.

Configuration

AArch32 System register PMINTENCLR bits [31:0] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[31:0\]](#).

AArch32 System register PMINTENCLR bits [31:0] are architecturally mapped to External register [PMU.PMINTENCLR_EL1\[31:0\]](#) [PMINTENCLR_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32 and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMINTENCLR are UNDEFINED.

Attributes

PMINTENCLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

[PMCCNTR](#) overflow interrupt request disable bit.

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request disable bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	When read, means that the PMEVCNTR<n> event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMEVCNTR<n> event counter interrupt request is enabled. When written, disables the PMEVCNTR<n> interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENCLR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMINTENCLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMINTENCLR;
elsif PSTATE.EL == EL3 then
    R[t] = PMINTENCLR;

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            PMINTENCLR = R[t];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                PMINTENCLR = R[t];
    elsif PSTATE.EL == EL3 then
        PMINTENCLR = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMINTENSET, Performance Monitors Interrupt Enable Set register

The PMINTENSET characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR](#), and the event counters [PMEVCNTR<n>](#). Reading the register shows which overflow interrupt requests are enabled.

PMINTENSET is used in conjunction with the [PMINTENCLR](#) register.

Configuration

AArch32 System register PMINTENSET bits [31:0] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[31:0\]](#).

AArch32 System register PMINTENSET bits [31:0] are architecturally mapped to External register [PMU.PMINTENSET_EL1\[31:0\]](#)[PMINTENSET_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32 and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMINTENSET are UNDEFINED.

Attributes

PMINTENSET is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

[PMCCNTR](#) overflow interrupt request enable bit.

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request enable bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	When read, means that the PMEVCNTR<n> event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMEVCNTR<n> event counter interrupt request is enabled. When written, enables the PMEVCNTR<n> interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENSET

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMINTENSET;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMINTENSET;
elsif PSTATE.EL == EL3 then
    R[t] = PMINTENSET;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            PMINTENSET = R[t];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                PMINTENSET = R[t];
    elsif PSTATE.EL == EL3 then
        PMINTENSET = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMMIR, Performance Monitors Machine Identification Register

The PMMIR characteristics are:

Purpose

Describes Performance Monitors parameters specific to the implementation to software.

Configuration

This register is present only when EL1 is capable of using AArch32 and FEAT_PMUv3p4 is implemented. Otherwise, direct accesses to PMMIR are UNDEFINED.

Attributes

PMMIR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	EDGE	THWIDTH	THWIDTH	BUS_WIDTH	BUS_WIDTH	BUS_SLOTS	BUS_SLOTS	SLOTS																							

Bits [31:**28**24]

Reserved, RES0.

EDGE, bits [27:24]

PMU event edge detection. Indicates implementation of the FEAT_PMUv3_EDGE feature.

EDGE	Meaning
0b0000	FEAT_PMUv3_EDGE is not implemented.
0b0001	FEAT_PMUv3_EDGE is implemented.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

THWIDTH, bits [23:20]

[PMEVTYPER<n>_ELO](#).TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the [PMEVTYPER<n>_ELO](#).TH field.

THWIDTH	Meaning
0b0000	FEAT_PMUv3_TH is not implemented.
0b0001	1 bit. PMEVTYPER<n>_EL0.TH[11:1] are RES0.
0b0010	2 bits. PMEVTYPER<n>_EL0.TH[11:2] are RES0.
0b0011	3 bits. PMEVTYPER<n>_EL0.TH[11:3] are RES0.
0b0100	4 bits. PMEVTYPER<n>_EL0.TH[11:4] are RES0.
0b0101	5 bits. PMEVTYPER<n>_EL0.TH[11:5] are RES0.
0b0110	6 bits. PMEVTYPER<n>_EL0.TH[11:6] are RES0.
0b0111	7 bits. PMEVTYPER<n>_EL0.TH[11:7] are RES0.
0b1000	8 bits. PMEVTYPER<n>_EL0.TH[11:8] are RES0.
0b1001	9 bits. PMEVTYPER<n>_EL0.TH[11:9] are RES0.
0b1010	10 bits. PMEVTYPER<n>_EL0.TH[11:10] are RES0.
0b1011	11 bits. PMEVTYPER<n>_EL0.TH[11] is RES0.
0b1100	12 bits.

All other values are reserved.

If FEAT_PMUv3_TH is not implemented, this field is zero.

Otherwise, the largest value that can be written to [PMEVTYPER<n>_EL0.TH](#) is $2^{(\text{PMMIR.THWIDTH})}$ minus one.

Note

[PMEVTYPER<n>_EL0.TH](#) cannot be accessed through [PMEVTYPER<n>](#).

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_WIDTH, bits [19:16]

Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\text{Log}_2(\text{number of bytes})$, plus one.

BUS_WIDTH	Meaning
0b0000	The information is not available.
0b0011	Four bytes.
0b0100	8 bytes.
0b0101	16 bytes.
0b0110	32 bytes.
0b0111	64 bytes.
0b1000	128 bytes.
0b1001	256 bytes.
0b1010	512 bytes.
0b1011	1024 bytes.
0b1100	2048 bytes.

All other values are reserved.

Each transfer is up to this number of bytes. An access might be smaller than the bus width.

When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.

When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS.

If the bus count information is not available, this field will read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SLOTS, bits [7:0]

Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMMIR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMMIR;
    elseif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                R[t] = PMMIR;
    elseif PSTATE.EL == EL3 then
        R[t] = PMMIR;

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMOVSr, Performance Monitors Overflow Flag Status Register

The PMOVSr characteristics are:

Purpose

Contains the state of the overflow bit for the Cycle Count Register, [PMCCNTR](#), and each of the implemented event counters [PMEVCNTR<n>](#). Writing to this register clears these bits.

Configuration

AArch32 System register PMOVSr bits [31:0] are architecturally mapped to AArch64 System register [PMOVSCLR_EL0\[31:0\]](#).

AArch32 System register PMOVSr bits [31:0] are architecturally mapped to External register [PMU.PMOVSCLR_EL0\[31:0\]](#) [PMOVSCLR_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMOVSr are UNDEFINED.

Attributes

PMOVSr is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

Cycle counter overflow clear bit. Possible values are:

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.

[PMCR](#).LC controls whether an overflow is detected from unsigned overflow of [PMCCNTR](#)[31:0] or unsigned overflow of [PMCCNTR](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow clear bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n> has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMEVCNTR<n> has overflowed since this bit was last cleared. When written, clears the PMEVCNTR<n> overflow bit to 0.

If FEAT_PMUv3p5 is implemented, [MDCR_EL2.HLP](#), [HDCR.HLP](#), and [PMCR.LP](#) control whether an overflow is detected from unsigned overflow of [PMEVCNTR<n>](#)[31:0] or unsigned overflow of [PMEVCNTR<n>](#)[63:0]. [PMEVCNTR<n>](#)[63:32] cannot be accessed directly in AArch32 state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSr

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMOVS ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSr;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSr;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSr;
elsif PSTATE.EL == EL3 then
    R[t] = PMOVSr;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMOVS ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSr = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSr = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSr = R[t];
elsif PSTATE.EL == EL3 then
    PMOVSr = R[t];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMOVSSET, Performance Monitors Overflow Flag Status Set register

The PMOVSSET characteristics are:

Purpose

Sets the state of the overflow bit for the Cycle Count Register, [PMCCNTR](#), and each of the implemented event counters [PMEVCNTR<n>](#).

Configuration

AArch32 System register PMOVSSET bits [31:0] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[31:0\]](#).

AArch32 System register PMOVSSET bits [31:0] are architecturally mapped to External register [PMU.PMOVSSET_EL0\[31:0\]](#)[PMOVSSET_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMOVSSET are UNDEFINED.

Attributes

PMOVSSET is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

C, bit [31]

Cycle counter overflow set bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.

[PMCR](#).LC controls whether an overflow is detected from unsigned overflow of [PMCCNTR](#)[31:0] or unsigned overflow of [PMCCNTR](#)[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow set bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are RAZ/WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	When read, means that PMEVCNTR<n> has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMEVCNTR<n> has overflowed since this bit was last . When written, sets the PMEVCNTR<n> overflow bit to 1.

If FEAT_PMUv3p5 is implemented, [MDCR_EL2.HLP](#), [HDCR.HLP](#), and [PMCR.LP](#) control whether an overflow is detected from unsigned overflow of [PMEVCNTR<n>](#)[31:0] or unsigned overflow of [PMEVCNTR<n>](#)[63:0]. [PMEVCNTR<n>](#)[63:32] cannot be accessed directly in AArch32 state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSSET

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMOVS ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSSSET;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSSSET;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMOVSSSET;
elsif PSTATE.EL == EL3 then
    R[t] = PMOVSSSET;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b011

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMOVS ==
'1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSSSET = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSSSET = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMOVSSSET = R[t];
elsif PSTATE.EL == EL3 then
    PMOVSSSET = R[t];

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMSELR, Performance Monitors Event Counter Selection Register

The PMSELR characteristics are:

Purpose

Selects the current event counter [PMEVCNTR<n>](#) or the cycle counter, CCNT.

PMSELR is used in conjunction with [PMXEVTYPER](#) to determine the event that increments a selected event counter, and the modes and states in which the selected counter increments.

It is also used in conjunction with [PMXVCNTR](#), to determine the value of a selected event counter.

Configuration

AArch32 System register PMSELR bits [31:0] are architecturally mapped to AArch64 System register [PMSELR_ELO\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMSELR are UNDEFINED.

Attributes

PMSELR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																										SEL					

Bits [31:5]

Reserved, RES0.

SEL, bits [4:0]

Selects event counter, [PMEVCNTR<n>](#), where n is the value held in this field. This value identifies which event counter is accessed when a subsequent access to [PMXEVTYPER](#) or [PMXVCNTR](#) occurs.

This field can take any value from 0 (0b00000) to (PMCR.N)-1, or 31 (0b11111).

When PMSELR.SEL is 0b11111, it selects the cycle counter and:

- A read of the [PMXEVTYPER](#) returns the value of [PMCCFILTR](#).
- A write of the [PMXEVTYPER](#) writes to [PMCCFILTR](#).
- A read or write of [PMXVCNTR](#) has CONSTRAINED UNPREDICTABLE effects. For more information, see [PMXVCNTR](#).

For more information about the results of accesses to event counters, see [PMXEVTYPER](#) and [PMXVCNTR](#).

For more information about the number of counters accessible at each Exception level, see [HDCR](#).HPMN and [MDCR_EL2](#).HPMN.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMSELR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b101


```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMSELR_EL0
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMSELR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMSELR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMSELR;
elsif PSTATE.EL == EL3 then
    R[t] = PMSELR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b101

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMSELR_EL0
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSELR = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSELR = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSELR = R[t];
elsif PSTATE.EL == EL3 then
    PMSELR = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMSWINC, Performance Monitors Software Increment register

The PMSWINC characteristics are:

Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW_INCR'.

Configuration

AArch32 System register PMSWINC bits [31:0] are architecturally mapped to AArch64 System register [PMSWINC_EL0\[31:0\]](#).

AArch32 System register PMSWINC bits [31:0] are architecturally mapped to External register [PMU.PMSWINC_EL0\[31:0\]](#)[PMSWINC_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMSWINC are UNDEFINED.

Attributes

PMSWINC is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Bit [31]

Reserved, RES0.

P<n>, bit [n], for n = 30 to 0

Event counter software increment bit for [PMEVCNTR<n>](#).

If N is less than 31, then bits [30:N] are WI. When EL2 is implemented and enabled in the current Security state, in EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64, or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

P<n>	Meaning
0b0	No action. The write to this bit is ignored.
0b1	If PMEVCNTR<n> is enabled and configured to count the software increment event, increments PMEVCNTR<n> by 1. If PMEVCNTR<n> is disabled, or not configured to count the software increment event, the write to this bit is ignored.

Accessing PMSWINC

Accesses to this register use the following encodings in the System register encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1100	0b100

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<SW,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<SW,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMSWINC_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSWINC = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSWINC = R[t];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMSWINC = R[t];
elsif PSTATE.EL == EL3 then
    PMSWINC = R[t];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMUSERENR, Performance Monitors User Enable Register

The PMUSERENR characteristics are:

Purpose

Enables or disables EL0 ~~User mode~~ access to the Performance Monitors.

Configuration

AArch32 System register PMUSERENR bits [31:0] are architecturally mapped to AArch64 System register [PMUSERENR_ELO\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3 is implemented. Otherwise, direct accesses to PMUSERENR are UNDEFINED.

Attributes

PMUSERENR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								TID	RES0	CR	SW	CR	EN	SW	EN

Bits [31:74]

Reserved, RES0.

TID, bit [6]

When FEAT_PMuV3p9 is implemented:

Trap ID registers. Traps EL0 read access to common event identification registers.

TID	Meaning
0b0	Accesses to PMCEID<n> are not trapped by this mechanism.
0b1	EL0 read accesses to PMCEID<n> are trapped.

The register accesses affected by this control are:

- MRC reads of [PMCEID0](#), [PMCEID1](#), [PMCEID2](#), and [PMCEID3](#).

When trapped, reads are UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:4]

Reserved, RES0.

ER, bit [3]

Event counters counter Readread enable.trap control:

When PMUSERENR.EN is 0, PMUSERENR.ER enables EL0 reads of the event counters and EL0 reads and writes of the select register.

ER	Meaning
0b0	EL0 reads of the event counters and EL0 reads and writes of the select register are disabled, unless enabled by PMUSERENR.EN. PMXVCNTR and PMEVCNTR<n>, and EL0 RW access to the PMSELR, are trapped to Undefined mode if PMUSERENR.EN is also 0.
0b1	EL0 Overrides reads PMUSERENR.EN of the event counters and EL0 enables reads RO and access writes of the select register are enabled, unless trapped by another control. to PMXVCNTR and PMEVCNTR<n>, and RW access to PMSELR.

The register accesses affected by this control are:

- MRC reads of PMEVCNTR<n> and PMXVCNTR.
- MRC and MCR accesses to PMSELR.

When disabled, reads and writes are UNDEFINED.

This field is ignored by the PE when PMUSERENR.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN0 value..

CR, bit [2]

Cycle counter Readread enable.trap control:

When PMUSERENR.EN is 0, PMUSERENR.CR enables EL0 reads of the cycle counter.

CR	Meaning
0b0	EL0 reads of the cycle counter are disabled, unless enabled by PMUSERENR.EN. PMCCNTR are trapped to Undefined mode if PMUSERENR.EN is also 0.
0b1	EL0 Overrides reads PMUSERENR.EN of and the enables cycle access counter are enabled, unless trapped by another control. to PMCCNTR.

The register accesses affected by this control are:

- MRC reads of PMCCNTR.
- MRRC reads of PMCCNTR.

When disabled, reads are UNDEFINED.

This field is ignored by the PE when PMUSERENR.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN0 value..

SW, bit [1]

Software increment register write Write trap enable.control:

When PMUSERENR.EN is 0, PMUSERENR.SW enables EL0 writes to the Software increment register.

SW	Meaning
0b0	EL0 writes to the Software increment register are disabled, unless enabled by PMUSERENR.EN. PMSWINC are trapped to Undefined mode if PMUSERENR.EN is also 0.
0b1	EL0 Overrides writes PMUSERENR.EN to and the enables Software access increment register are enabled, unless trapped by another control to PMSWINC.

The register accesses affected by this control are:

- MCR writes to PMSWINC.

When disabled, writes are UNDEFINED.

This field is ignored by the PE when PMUSERENR.EN == 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN0 value.

EN, bit [0]

- PMCR, PMOVSr, PMSELR, PMCEID0, PMCEID1, PMCCNTR, PMXEVTYPER, PMXVCNTR, PMCNTENSET, PMCNTENCLR, PMOVSSET, PMEVCNTR<n>, PMEVTYPER<n>, PMCCFILTR, PMSWINC.
- If FEAT_PMUv3p1 is implemented, PMCEID2, and PMCEID3.
- If FEAT_PMUv3p4 is implemented, PMMIR.

Enable. Traps Enables EL0 read/write accesses to the Performance Monitors registers to PMU Undefined registers. mode, as follows:

EN	Meaning
0b0	While at EL0, accesses to PMU the specified registers at EL0 are trapped to Undefined mode, unless enabled overridden by one of PMUSERENR.{ER, CR, SW}.
0b1	EL0 While accesses at to EL0, PMU software registers can access enabled, all unless of trapped the by specified another control registers.

The register accesses affected by this control are:

- MRC or MCR accesses to PMCCFILTR, PMCCNTR, PMCNTENCLR, PMCNTENSET, PMCR, PMEVCNTR<n>, PMEVTYPER<n>, PMOVSr, PMOVSSET, PMSELR, PMXVCNTR, and PMXEVTYPER.
- MRC reads of the following registers:
 - PMCEID0 and PMCEID1.
 - If FEAT_PMUv3p1 is implemented, PMCEID2 and PMCEID3.
- MCR writes to PMSWINC.
- MRRC or MCRR accesses to PMCCNTR.

When trapped, reads and writes are UNDEFINED.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN0 value.

Accessing PMUSERENR

When FEAT_PMUv3p9 is implemented and EL1 is using AArch64, PMUSERENR_EL0 contains additional controls that affect the behavior of this register.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
--------	------	-----	-----	------

0b1111	0b000	0b1001	0b1110	0b000
--------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMUSERENR_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            R[t] = PMUSERENR;
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                R[t] = PMUSERENR;
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                R[t] = PMUSERENR;
    elsif PSTATE.EL == EL3 then
        R[t] = PMUSERENR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1110	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            PMUSERENR = R[t];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
            else
                PMUSERENR = R[t];
    elsif PSTATE.EL == EL3 then
        PMUSERENR = R[t];

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMXEVCNTR, Performance Monitors Selected Event Count Register

The PMXEVCNTR characteristics are:

Purpose

Reads or writes the value of the selected event counter, [PMEVCNTR<n>](#). [PMSELR](#).SEL determines which event counter is selected.

Configuration

AArch32 System register PMXEVCNTR bits [31:0] are architecturally mapped to AArch64 System register [PMEVCNTR_EL0\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMXEVCNTR are UNDEFINED.

Attributes

PMXEVCNTR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMEVCNTR<n>																															

PMEVCNTR<n>, bits [31:0]

Value of the selected event counter, [PMEVCNTR<n>](#), where n is the value stored in [PMSELR](#).SEL.

If FEAT_PMUv3p5 is implemented, the event counter is 64 bits and only the least-significant part of the event counter is accessible in AArch32 state:

- Reads from PMXEVCNTR return bits [31:0] of the counter.
- Writes to PMXEVCNTR update bits [31:0] and leave bits [63:32] unchanged.
- There is no means to access bits [63:32] directly from AArch32 state.
- If the implementation does not support AArch64, bits [63:32] are not required to be implemented.

If FEAT_PMUv3p5 is not implemented, the event counter is 32 bits.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMXEVCNTR

If FEAT_FGT is implemented and [PMSELR](#).SEL is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMEVCNTR](#) is as follows:

- If [PMSELR](#).SEL selects an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and [PMSELR.SEL](#) is greater than or equal to the number of accessible event counters, then reads and writes of [PMXEVCNTR](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP
- Accesses to the register behave as if [PMSELR.SEL](#) has an UNKNOWN value less than the number of event counters accessible at the current Exception level and Security state.
- If EL2 is implemented and enabled in the current Security state, and [PMSELR.SEL](#) is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR.{ER,EN}](#) or [PMUSERENR_EL0.{ER,EN}](#).

If EL2 is implemented and enabled in the current Security state, at EL0 and EL1:

- If EL2 is using AArch32, [HDCR.HPMN](#) identifies the number of accessible event counters.
- If EL2 is using AArch64, [MDCR_EL2.HPMN](#) identifies the number of accessible event counters.

Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [HDCR.HPMN](#) and [MDCR_EL2.HPMN](#).

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b010

```

if UInt(PMSELR.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVCNTR[UInt(PMSELR.SEL)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;

```



```

        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVCNTR[UInt(PMSELR.SEL)];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        R[t] = PMEVCNTR[UInt(PMSELR.SEL)];
elseif PSTATE.EL == EL3 then
    R[t] = PMEVCNTR[UInt(PMSELR.SEL)];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b010

```

if UInt(PMSELR.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[UInt(PMSELR.SEL)] = R[t];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) >= AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;

```

```

        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[UInt(PMSELR.SEL)] = R[t];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[UInt(PMSELR.SEL)] = R[t];
elseif PSTATE.EL == EL3 then
    PMEVCNTR[UInt(PMSELR.SEL)] = R[t];

```

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMXEVTYPER, Performance Monitors Selected Event Type Register

The PMXEVTYPER characteristics are:

Purpose

When [PMSELR.SEL](#) selects an event counter, this accesses a [PMEVTYPER<n>](#) register. When [PMSELR.SEL](#) selects the cycle counter, this accesses [PMCCFILTER](#).

Configuration

AArch32 System register PMXEVTYPER bits [31:0] are architecturally mapped to AArch64 System register [PMXEVTYPER_ELO\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMXEVTYPER are UNDEFINED.

Attributes

PMXEVTYPER is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Event type register or PMCCFILTER																															

Bits [31:0]

Event type register or [PMCCFILTER](#).

When [PMSELR.SEL](#) == 31, this register accesses [PMCCFILTER](#).

Otherwise, this register accesses [PMEVTYPER<n>](#) where n is the value in [PMSELR.SEL](#).

Accessing PMXEVTYPER

If FEAT_FGT is implemented, and [PMSELR.SEL](#) is not 31 and is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of [PMXEVTYPER](#) is as follows:

- If [PMSELR.SEL](#) selects an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented, and [PMSELR.SEL](#) is not 31 and is greater than or equal to the number of accessible event counters, then reads and writes of [PMXEVTYPER](#) are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP
- Accesses to the register behave as if [PMSELR.SEL](#) has an UNKNOWN value less than the number of event counters accessible at the current Exception level and Security state.
- Accesses to the register behave as if [PMSELR.SEL](#) is 31.
- If EL2 is implemented and enabled in the current Security state, and [PMSELR.SEL](#) is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by [PMUSERENR.EN](#) or [PMUSERENR_EL0.EN](#).

If EL2 is implemented and enabled in the current Security state, at EL0 and EL1:

- If EL2 is using AArch32, [HDCR.HPMN](#) identifies the number of accessible event counters.
- If EL2 is using AArch64, [MDCR_EL2.HPMN](#) identifies the number of accessible event counters.

Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see [HDCR.HPMN](#) and [MDCR_EL2.HPMN](#).

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b001

```

if UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >=
AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        if UInt(PMSELR.SEL) == 31 then
            R[t] = PMCCFILTR;
        else
            R[t] = PMEVTYPER[UInt(AArch32-PMSELR.SEL)];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >=
AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL2) then
            AArch32.TakeHypTrapException(0x03);

```

```

    else
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            if UInt(PMSELR.SEL) == 31 then
                R[t] = PMCCFILTR;
            else
                R[t] = PMEVTYPER[UInt(AArch32-PMSELR.SEL)];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            if UInt(PMSELR.SEL) == 31 then
                R[t] = PMCCFILTR;
            else
                R[t] = PMEVTYPER[UInt(AArch32-PMSELR.SEL)];
    elsif PSTATE.EL == EL3 then
        if UInt(PMSELR.SEL) == 31 then
            R[t] = PMCCFILTR;
        else
            R[t] = PMEVTYPER[UInt(AArch32-PMSELR.SEL)];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1001	0b1101	0b001

```

if UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif !ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elsif ELUsingAArch32(EL1) && PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T9 == '1'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >=
AArch32.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        elsif ELUsingAArch32(EL1) then
            AArch32.TakeHypTrapException(0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        if UInt(PMSELR.SEL) == 31 then
            PMCCFILTR = R[t];
        else
            PMEVTYPER[UInt(AArch32-PMSELR.SEL)] = R[t];
    elsif PSTATE.EL == EL1 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T9 == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T9 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.TPM == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.TPM == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && UInt(PMSELR.SEL) != 31 && UInt(PMSELR.SEL) >=
AArch32.GetNumEventCountersAccessible() then
            if !IsFeatureImplemented(FEAT_FGT) then
                ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
            elsif ELUsingAArch32(EL2) then
                AArch32.TakeHypTrapException(0x03);

```



```

    else
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            if UInt(PMSELR.SEL) == 31 then
                PMCCFILTR = R[t];
            else
                PMEVTYPER[UInt(AArch32-PMSELR.SEL)] = R[t];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x03);
        else
            if UInt(PMSELR.SEL) == 31 then
                PMCCFILTR = R[t];
            else
                PMEVTYPER[UInt(AArch32-PMSELR.SEL)] = R[t];
    elsif PSTATE.EL == EL3 then
        if UInt(PMSELR.SEL) == 31 then
            PMCCFILTR = R[t];
        else
            PMEVTYPER[UInt(AArch32-PMSELR.SEL)] = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

SCTLR, System Control Register

The SCTLR characteristics are:

Purpose

Provides the top level control of the system, including its memory system.

Configuration

AArch32 System register SCTLR bits [31:0] are architecturally mapped to AArch64 System register [SCTLR_EL1\[31:0\]](#).

This register is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to SCTLR are UNDEFINED.

Some bits in the register are read-only. These bits relate to non-configurable features of an implementation, and are provided for compatibility with previous versions of the architecture.

Attributes

SCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
DSSBS	TE	AF	TRE	RES0	EE	RES0	SPAN	RES1	RES0	UWXN	WXN	nTW	RES0	nTW	RES0	V	I	RES1	EnR	CTX	RES0	SED	ITD	UNK	

DSSBS, bit [31] When FEAT_SSBS is implemented:

Default PSTATE.SSBS value on Exception Entry. The defined values are:

DSSBS	Meaning
0b0	PSTATE.SSBS is set to 0 on an exception to any mode in this security state except Hyp mode
0b1	PSTATE.SSBS is set to 1 on an exception to any mode in this security state except Hyp mode

Note

When EL3 is implemented and is using AArch32, this bit is banked between the two Security states.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Otherwise:

Reserved, RES0.

TE, bit [30]

T32 Exception Enable. This bit controls whether exceptions to an Exception level that is executing at PL1 are taken to A32 or T32 state:

TE	Meaning
0b0	Exceptions, including reset, taken to A32 state.
0b1	Exceptions, including reset, taken to T32 state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

AFE, bit [29]

Access Flag Enable. When using the Short-descriptor translation table format for the PL1&0 translation regime, this bit enables use of the AP[0] bit in the translation descriptors as the Access flag, and restricts access permissions in the translation descriptors to the simplified model.

AFE	Meaning
0b0	In the Translation table descriptors, AP[0] is an access permissions bit. The full range of access permissions is supported. No Access flag is implemented.
0b1	In the Translation table descriptors, AP[0] is the Access flag. Only the simplified model for access permissions is supported.

When using the Long-descriptor translation table format, the VMSA behaves as if this bit is set to 1, regardless of the value of this bit.

The AFE bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

TRE, bit [28]

TEX remap enable. This bit enables remapping of the TEX[2:1] bits in the PL1&0 translation regime for use as two translation table bits that can be managed by the operating system. Enabling this remapping also changes the scheme used to describe the memory region attributes in the VMSA.

TRE	Meaning
0b0	TEX remap disabled. TEX[2:0] are used, with the C and B bits, to describe the memory region attributes.
0b1	TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by the operating system. The TEX[0], C, and B bits are used to describe the memory region attributes, with the MMU remap registers.

When the value of [TTBCR.EAE](#) is 1, this bit is RES1.

The TRE bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Bits [27:26]

Reserved, RES0.

EE, bit [25]

The value of the PSTATE.E bit on branch to an exception vector or coming out of reset, and the endianness of stage 1 translation table walks in the PL1&0 translation regime.

EE	Meaning
0b0	Little-endian. PSTATE.E is cleared to 0 on taking an exception or coming out of reset. Stage 1 translation table walks in the PL1&0 translation regime are little-endian.
0b1	Big-endian. PSTATE.E is set to 1 on taking an exception or coming out of reset. Stage 1 translation table walks in the PL1&0 translation regime are big-endian.

If an implementation does not provide Big-endian support for data accesses at Exception levels higher than EL0, this bit is RES0.

If an implementation does not provide Little-endian support for data accesses at Exception levels higher than EL0, this bit is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Bit [24]

Reserved, RES0.

SPAN, bit [23]

When FEAT_PAN is implemented:

Set Privileged Access Never, on taking an exception to EL1 from either Secure or Non-secure state, or to EL3 from Secure state when EL3 is using AArch32.

SPAN	Meaning
0b0	PSTATE.PAN is set to 1 in the following situations: <ul style="list-style-type: none"> In Non-secure state, on taking an exception to EL1. In Secure state, when EL3 is using AArch64, on taking an exception to EL1. In Secure state, when EL3 is using AArch32, on taking an exception to EL3.
0b1	The value of PSTATE.PAN is left unchanged on taking an exception to EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

Bit [22]

Reserved, RES1.

Bit [21]

Reserved, RES0.

UWXN, bit [20]

Unprivileged write permission implies PL1 XN (Execute-never). This bit can force all memory regions that are writable at PL0 to be treated as XN for accesses from software executing at PL1.

UWXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable at PL0 forced to XN for accesses from software executing at PL1.

The UWXN bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

WXN, bit [19]

Write permission implies XN (Execute-never). For the PL1&0 translation regime, this bit can force all memory regions that are writable to be treated as XN.

WXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable in the PL1&0 translation regime is forced to XN for accesses from software executing at PL1 or PL0.

This bit applies only when SCTLR.M bit is set.

The WXN bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

nTWE, bit [18]

Traps EL0 execution of WFE instructions to Undefined mode.

nTWE	Meaning
0b0	Any attempt to execute a WFE instruction at EL0 is trapped to Undefined mode, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

The attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

Traps EL0 execution of WFI instructions to Undefined mode.

nTWI	Meaning
0b0	Any attempt to execute a WFI instruction at EL0 is trapped to Undefined mode, if the instruction would otherwise have caused the PE to enter a low-power state.
0b1	This control does not cause any instructions to be trapped.

The attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

Note

Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Bits [15:14]

Reserved, RES0.

V, bit [13]

Vectors bit. This bit selects the base address of the exception vectors for exceptions taken to a PE mode other than Monitor mode or Hyp mode:

V	Meaning
0b0	Normal exception vectors. Base address is held in VBAR .
0b1	High exception vectors (Hivecs), base address 0xFFFF0000. This base address cannot be remapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

I, bit [12]

Instruction access Cacheability control, for accesses at EL1 and EL0:

I	Meaning
0b0	All instruction access to Normal memory from PL1 and PL0 are Non-cacheable for all levels of instruction and unified cache. If the value of SCTLR.M is 0, instruction accesses from stage 1 of the PL1&0 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
0b1	All instruction access to Normal memory from PL1 and PL0 can be cached at all levels of instruction and unified cache. If the value of SCTLR.M is 0, instruction accesses from stage 1 of the PL1&0 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

Instruction accesses to Normal memory from EL1 and EL0 are Cacheable regardless of the value of the SCTLR.I bit if either:

- EL2 is using AArch32 and the value of [HCR.DC](#) is 1.
- EL2 is using AArch64 and the value of [HCR_EL2.DC](#) is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Bit [11]

Reserved, RES1.

EnRCTX, bit [10]
When FEAT_SPECRES is implemented:

Enable EL0 access to the following AArch32 System CPFRCTX, instructions: DVPRCTX, and CPPRCTX instructions.

- [CFPRCTX](#).
- [DVPRCTX](#).
- [CPPRCTX](#).
- If FEAT_SPECRES2 is implemented, [COSPRCTX](#).

EnRCTX	Meaning
0b0	EL0 access to these instructions is disabled, and these instructions are trapped to EL1.
0b1	EL0 access to these instructions is enabled.

Note

When EL3 is implemented and is using AArch32, this bit is banked between the two Security states.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [9]

Reserved, RES0.

SED, bit [8]

SETEND instruction disable. Disables SETEND instructions at PL0 and PL1.

SED	Meaning
0b0	SETEND instruction execution is enabled at PL0 and PL1.
0b1	SETEND instructions are UNDEFINED at PL0 and PL1.

If the implementation does not support mixed-endian operation at any Exception level, this bit is RES1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

ITD, bit [7]

IT Disable. Disables some uses of IT instructions at PL1 and PL0.

ITD	Meaning
0b0	All IT instruction functionality is enabled at PL1 and PL0.
0b1	Any attempt at PL1 or PL0 to execute any of the following is UNDEFINED: <ul style="list-style-type: none"> All encodings of the IT instruction with hw1[3:0]!=1000. All encodings of the subsequent instruction with the following values for hw1: <ul style="list-style-type: none"> 11xxxxxxxxxxxx: All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM. 1011xxxxxxxxxxxx: All instructions in 'Miscellaneous 16-bit instructions'. 10100xxxxxxxxxxx: ADD Rd, PC, #imm 01001xxxxxxxxxxx: LDR Rd, [PC, #imm] 0100x1xxx1111xxx: ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC. 010001xx1xxxx111: ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers unpredictable cases with BLX Rn. <p>These instructions are always UNDEFINED, regardless of whether they would pass or fail the condition code check that applies to them as a result of being in an IT block.</p> <p>It is IMPLEMENTATION DEFINED whether the IT instruction is treated as:</p> <ul style="list-style-type: none"> A 16-bit instruction, that can only be followed by another 16-bit instruction. The first half of a 32-bit instruction. <p>This means that, for the situations that are UNDEFINED, either the second 16-bit instruction or the 32-bit instruction is UNDEFINED.</p> <p>An implementation might vary dynamically as to whether IT is treated as a 16-bit instruction or the first half of a 32-bit instruction.</p>

If an instruction in an active IT block that would be disabled by this field sets this field to 1 then behavior is CONSTRAINED UNPREDICTABLE. For more information see 'Changes to an ITD control by an instruction in an IT block'.

ITD is optional, but if it is implemented in the SCTLR then it must also be implemented in the [SCTLR_EL1](#), [SCTLR_EL2](#), and [HSCTLR](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

When an implementation does not implement ITD, access to this field is **RAZ/WI**.

UNK, bit [6]

Writes to this bit are IGNORED. Reads of this bit return an UNKNOWN value.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

CP15BEN, bit [5]

System instruction memory barrier enable. Enables accesses to the DMB, DSB, and ISB System instructions in the (coproc==0b1111) encoding space from PL1 and PL0:

CP15BEN	Meaning
0b0	PL0 and PL1 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
0b1	PL0 and PL1 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

CP15BEN is optional, but if it is implemented in the SCTLR then it must also be implemented in the [SCTLR_EL1](#), [SCTLR_EL2](#), and [HSCTLR](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

When an implementation does not implement CP15BEN, access to this field is **RAO/WI**.

LSMAOE, bit [4]

When FEAT_LSMAOC is implemented:

Load Multiple and Store Multiple Atomicity and Ordering Enable.

LSMAOE	Meaning
0b0	For all memory accesses at EL1 or EL0, A32 and T32 Load Multiple and Store Multiple can have an interrupt taken during the sequence memory accesses, and the memory accesses are not required to be ordered.
0b1	The ordering and interrupt behavior of A32 and T32 Load Multiple and Store Multiple at EL1 or EL0 is as defined for Armv8.0.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Otherwise:

Reserved, RES1.

nTLSMD, bit [3]

When FEAT_LSMAOC is implemented:

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory.

nTLSMD	Meaning
0b0	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL1 or EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are trapped and generate a stage 1 Alignment fault.
0b1	All memory accesses by A32 and T32 Load Multiple and Store Multiple at EL1 or EL0 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are not trapped.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Otherwise:

Reserved, RES1.

C, bit [2]

Cacheability control, for data accesses at EL1 and EL0:

C	Meaning
0b0	All data access to Normal memory from PL1 and PL0, and all accesses to the PL1&0 stage 1 translation tables, are Non-cacheable for all levels of data and unified cache.
0b1	All data access to Normal memory from PL1 and PL0, and all accesses to the PL1&0 stage 1 translation tables, can be cached at all levels of data and unified cache.

The PE ignores SCTLR.C, ~~for Non-secure state~~ and data accesses to Normal memory from EL1 and EL0 are Cacheable, if either:

- EL2 is using AArch32 and the value of [HCR.DC](#) is 1.
- EL2 is using AArch64 and the value of [HCR_EL2.DC](#) is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at PL1 and PL0:

A	Meaning
0b0	Alignment fault checking disabled when executing at PL1 or PL0. Instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, do not check that the address being accessed is aligned to the size of the data element(s) being accessed.
0b1	Alignment fault checking enabled when executing at PL1 or PL0. All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have an alignment check regardless of the value of the A bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

M, bit [0]

MMU enable for EL1 and EL0 stage 1 address translation. Possible values of this bit are:

M	Meaning
0b0	EL1 and EL0 stage 1 address translation disabled. See the SCTLR.I field for the behavior of instruction accesses to Normal memory.
0b1	EL1 and EL0 stage 1 address translation enabled.

~~TheIn the Non-secure state the~~ PE behaves as if the value of the SCTLR.M field is 0 for all purposes other than returning the value of a direct read of the field if either:

- EL2 is using AArch32 and the value of [HCR.{DC, TGE}](#) is not {0, 0}.
- EL2 is using AArch64 and the value of [HCR_EL2.{DC, TGE}](#) is not {0, 0}.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SCTLR

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TRVM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TRVM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        R[t] = SCTLR_NS;
    else
        R[t] = SCTLR;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        R[t] = SCTLR_NS;
    else
        R[t] = SCTLR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        R[t] = SCTLR_S;
    else
        R[t] = SCTLR_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TVM == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TVM == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        SCTLR_NS = R[t];
    else
        SCTLR = R[t];
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        SCTLR_NS = R[t];
    else
        SCTLR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE == HIGH then
        UNDEFINED;
    elsif SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        if SCR.NS == '0' then
            SCTLR_S = R[t];
        else
            SCTLR_NS = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TLBIALL, TLB Invalidate All

The TLBIALL characteristics are:

Purpose

Invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk. The entries that are invalidated are as follows:

- If executed at EL1, all entries that:
 - Would be required for the EL1&0 translation regime.
 - Match the current VMID, if EL2 is implemented and enabled in the current Security state.
- If executed in Secure state when EL3 is using AArch32, all entries that would be required for the Secure PL1&0 translation regime.
- If executed at EL2, and if EL2 is enabled in the current Security state, the stage 1 or stage 2 translation table entries that would be required for the PL1&0 translation regime and matches the current VMID.

The invalidation only applies to the PE that executes this System instruction.

Configuration

This instruction is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to TLBIALL are UNDEFINED.

Attributes

TLBIALL is a 32-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Rt> is ignored.

Executing the TLBIALL instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1000	0b0111	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_ExcludeXS);
        else
            AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_ExcludeXS);
            else
                AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_AllAttr);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBI_AllAttr);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_ALL(SecurityStateAtEL(EL3), Regime_EL30, Shareability_NSH, TLBI_ExcludeXS);

```

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBIASID, TLB Invalidate by ASID match

The TLBIASID characteristics are:

Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- If EL2 is implemented and enabled in the current Security state, the entry would be used with the current VMID.

From the entries that match these requirements, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this System instruction.

Configuration

This instruction is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to TLBIASID are UNDEFINED.

Attributes

TLBIASID is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								ASID							

Bits [31:8]

Reserved, RES0.

ASID, bits [7:0]

ASID value to match. Any TLB entries for non-global pages that match the ASID values will be affected by this System instruction.

Executing the TLBIASID instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
--------	------	-----	-----	------

0b1111	0b000	0b1000	0b0111	0b010
--------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_ExcludeXS, R[t]);
        else
            AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr, R[t]);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr, R[t]);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_ExcludeXS, R[t]);
            else
                AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH, TLBI_AllAttr,
        R[t]);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_ASID(SecurityStateAtEL(EL3), Regime_EL30, VMID_NONE, Shareability_NSH,
        TLBI_AllAttr, R[t]);

```

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBIMVA, TLB Invalidate by VA

The TLBIMVA characteristics are:

Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented and enabled in the current Security state, the entry would be used with the current VMID.

From the entries that match these requirements, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this System instruction.

Configuration

This instruction is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to TLBIMVA are UNDEFINED.

Attributes

TLBIMVA is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VA												RES0				ASID															

VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Bits [11:8]

Reserved, RES0.

ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

Executing the TLBIMVA instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1000	0b0111	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, R[t]);
        else
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, R[t]);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, R[t]);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, R[t]);
            else
                AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH, TLBIlevel_Any,
        TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL30, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, R[t]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_ExcludeXS, R[t]);
        else
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, R[t]);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, R[t]);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_ExcludeXS, R[t]);
            else
                AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Any, TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH, TLBIlevel_Any,
        TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_VAA(SecurityStateAtEL(EL3), Regime_EL30, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, R[t]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBIMVAAL, TLB Invalidate by VA, All ASID, Last level

The TLBIMVAAL characteristics are:

Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- If EL2 is implemented and enabled in the current Security state, the entry would be used with the current VMID.

From the entries that match these requirements, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this System instruction.

Configuration

This instruction is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to TLBIMVAAL are UNDEFINED.

Note

This System instruction is not implemented in architecture versions before Armv8.

Attributes

TLBIMVAAL is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VA												RES0																			

VA, bits [31:12]

Virtual address to match. Any unlocked TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

Bits [11:0]

Reserved, RES0.

Executing the TLBIMVAAL instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1000	0b0111	0b111

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, R[t]);
        else
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, R[t]);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, R[t]);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, R[t]);
            else
                AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_VAA(SecurityStateAtEL(EL3), Regime_EL30, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, R[t]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TLBIMVAL, TLB Invalidate by VA, Last level

The TLBIMVAL characteristics are:

Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented and enabled in the current Security state, the entry would be used with the current VMID.

From the entries that match these requirements, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this System instruction.

Configuration

This instruction is present only when EL1 is capable of using AArch32. Otherwise, direct accesses to TLBIMVAL are UNDEFINED.

This System instruction is not implemented in architecture versions before Armv8.

Attributes

TLBIMVAL is a 32-bit System instruction.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VA												RES0				ASID															

VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Bits [11:8]

Reserved, RES0.

ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

Executing the TLBIMVAL instruction

Accesses to this instruction use the following encodings in the System instruction encoding space:

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1000	0b0111	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T8 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T8 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TTLB == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TTLB == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) && IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_ExcludeXS, R[t]);
        else
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, R[t]);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.FB == '1' then
            AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, R[t]);
        else
            if IsFeatureImplemented(FEAT_XS) && !ELUsingAArch32(EL2) && IsFeatureImplemented(FEAT_HCX)
            && IsHCRXEL2Enabled() && HCRX_EL2.FnXS == '1' then
                AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_ExcludeXS, R[t]);
            else
                AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
                TLBIlevel_Last, TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL2 then
        AArch32.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH, TLBIlevel_Last,
        TLBI_AllAttr, R[t]);
    elsif PSTATE.EL == EL3 then
        AArch32.TLBI_VA(SecurityStateAtEL(EL3), Regime_EL30, VMID_NONE, Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, R[t]);

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TPIDRURO, PL0 Read-Only Software Thread ID Register

The TPIDRURO characteristics are:

Purpose

Provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

Configuration

AArch32 System register TPIDRURO bits [31:0] are architecturally mapped to AArch64 System register [TPIDRRO_EL0\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to TPIDRURO are UNDEFINED.

Note

The PE never updates this register.

Attributes

TPIDRURO is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Thread ID																															

Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TPIDRURO

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
    IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.TPIDRRO_EL0
    == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    else
        R[t] = TPIDRURO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
            R[t] = TPIDRURO_NS;
        else
            R[t] = TPIDRURO;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            R[t] = TPIDRURO_NS;
        else
            R[t] = TPIDRURO;
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            R[t] = TPIDRURO_S;
        else
            R[t] = TPIDRURO_NS;
    
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
            TPIDRURO_NS = R[t];
        else
            TPIDRURO = R[t];
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            TPIDRURO_NS = R[t];
        else
            TPIDRURO = R[t];
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            TPIDRURO_S = R[t];
        else
            TPIDRURO_NS = R[t];
    
```

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TPIDRURW, PL0 Read/Write Software Thread ID Register

The TPIDRURW characteristics are:

Purpose

Provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

Configuration

AArch32 System register TPIDRURW bits [31:0] are architecturally mapped to AArch64 System register [TPIDR_EL0\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to TPIDRURW are UNDEFINED.

Note

The PE never updates this register.

Attributes

TPIDRURW is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Thread ID																															

Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing TPIDRURW

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0000	0b010

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
    IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.TPIDR_EL0
    == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    else
        R[t] = TPIDRURW;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        R[t] = TPIDRURW_NS;
    else
        R[t] = TPIDRURW;
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        R[t] = TPIDRURW_NS;
    else
        R[t] = TPIDRURW;
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        R[t] = TPIDRURW_S;
    else
        R[t] = TPIDRURW_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1101	0b0000	0b010

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && HSTR_EL2.T13 == '1'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL1) && HCR_EL2.<E2H,TGE> != '11' &&
    IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.TPIDR_EL0
    == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    else
        TPIDRURW = R[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T13 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T13 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        TPIDRURW_NS = R[t];
    else
        TPIDRURW = R[t];
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        TPIDRURW_NS = R[t];
    else
        TPIDRURW = R[t];
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        TPIDRURW_S = R[t];
    else
        TPIDRURW_NS = R[t];

```

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VDISR, Virtual Deferred Interrupt Status Register

The VDISR characteristics are:

Purpose

Records that an SError interrupt has been consumed by an ESB instruction.

Configuration

AArch32 System register VDISR bits [31:0] are architecturally mapped to AArch64 System register [VDISR_EL2\[31:0\]](#).

This register is present only when **EL1 is capable of using AArch32 and** FEAT_RAS is implemented. Otherwise, direct accesses to VDISR are UNDEFINED.

If EL2 is not implemented, then VDISR is RES0 from Monitor mode when SCR.NS == 1.

Attributes

VDISR is a 32-bit register.

Field descriptions

When TTBCR.EAE == 0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	RES0														AET	RES0	EXT	RES0	FS[4]	LPAE	RES0				FS[3:0]						

A, bit [31]

Set to 1 when an ESB instruction defers a virtual SError interrupt.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [30:16]

Reserved, RES0.

AET, bits [15:14]

The value copied from [VDESR.AET](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [13]

Reserved, RES0.

ExT, bit [12]

The value copied from [VDFSR.ExT](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [11]

Reserved, RES0.

FS, bits [10, 3:0]

Fault status code. Set to 0b10110 when an ESB instruction defers a virtual SError interrupt.

FS	Meaning
0b10110	Asynchronous SError interrupt.

All other values are reserved.

The FS field is split as follows:

- FS[4] is VDISR[10].
- FS[3:0] is VDISR[3:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

LPAE, bit [9]

Format.

Set to [TTBCR.EAE](#) when an ESB instruction defers a virtual SError interrupt.

LPAE	Meaning
0b0	Using the Short-descriptor translation table format.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [8:4]

Reserved, RES0.

When TTBCR.EAE == 1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	RES0										AET		RES0	ExT	RES0	LPAE	RES0		STATUS												

A, bit [31]

Set to 1 when an ESB instruction defers a virtual SError interrupt.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [30:16]

Reserved, RES0.

AET, bits [15:14]

The value copied from [VDFSR.AET](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bit [13]

Reserved, RES0.

ExT, bit [12]

The value copied from [VDFSR.ExT](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [11:10]

Reserved, RES0.

LPAE, bit [9]

Format.

Set to [TTBCR.EAE](#) when an ESB instruction defers a virtual SError interrupt.

LPAE	Meaning
0b1	Using the Long-descriptor translation table format.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Bits [8:6]

Reserved, RES0.

STATUS, bits [5:0]

Fault status code. Set to 0b010001 when an ESB instruction defers a virtual SError interrupt.

STATUS	Meaning
0b010001	Asynchronous SError interrupt.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing VDISR

Direct reads and writes of VDFSR are UNDEFINED if EL3 is implemented and using AArch32 in all Secure privileged modes other than Monitor mode.

An indirect write to VDISR made by an ESB instruction does not require an explicit synchronization operation for the value that is written to be observed by a direct read of [DISR](#) occurring in program order after the ESB instruction.

If EL2 is not implemented, then VDISR is RES0 from Monitor mode when [SCR.NS](#) == 1.

Accesses to this register use the following encodings in the System register encoding space:

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b1100	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    R[t] = VDISR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        R[t] = VDISR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b1100	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VDISR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        VDISR = R[t];

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.AM0 == '1' then
        R[t] = VDISR_EL2<31:0>;
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.AM0 == '1' then
        R[t] = VDISR;
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && !Halted() && SCR_EL3.EA == '1' then
        R[t] = Zeros(32);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) && !Halted() && SCR.EA == '1' then
        R[t] = Zeros(32);
    else
        R[t] = DISR;
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && !ELUsingAArch32(EL3) && !Halted() && SCR_EL3.EA == '1' then
        R[t] = Zeros(32);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) && !Halted() && SCR.EA == '1' then
        R[t] = Zeros(32);
    else
        R[t] = DISR;
elseif PSTATE.EL == EL3 then
    R[t] = DISR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b0001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T12 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T12 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.AM0 == '1' then
        VDISR_EL2 = R[t];
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.AM0 == '1' then
        VDISR = R[t];
    elseif HaveEL(EL3) && !ELUsingAArch32(EL3) && !Halted() && SCR_EL3.EA == '1' then
        return;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) && !Halted() && SCR.EA == '1' then
        return;
    else
        DISR = R[t];
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && !ELUsingAArch32(EL3) && !Halted() && SCR_EL3.EA == '1' then
        return;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) && !Halted() && SCR.EA == '1' then
        return;
    else
        DISR = R[t];
elseif PSTATE.EL == EL3 then
    DISR = R[t];

```

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

System Register index by instruction and encoding

Below are indexes for registers and operations accessed in the following ways:

For AArch32

- [MCR/MRC](#)
- [MCRR/MRRC](#)
- [MRS/MSR](#)
- [VMRS/VMSR](#)

For AArch64

- [AT](#)
- [BRB](#)
- [CFP](#)
- [CPP](#)
- [DC](#)
- [DVP](#)
- [IC](#)
- [MRS/MSR](#)
- [MRRS/MSRR](#)[TLBI](#)
- [TLBI](#)
- [TLBIP](#)

Registers and operations in AArch32

Accessed using MCR/MRC:

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1110	000	0000	0000	000	RO	DBGDIDR	DBGDIDR
1110	000	0000	0000	010	RW	DBGDTRRXext	DBGDTRRXext
1110	000	0000	0001	000	RO	DBGDSCRint	DBGDSCRint
1110	000	0000	0010	000	RW	DBGDCCINT	DBGDCCINT
1110	000	0000	0010	010	RW	DBGDSCRext	DBGDSCRext
1110	000	0000	0011	010	RW	DBGDTRTXext	DBGDTRTXext
1110	000	0000	0101	000	RO	DBGDTRRXint	DBGDTRRXint
1110	000	0000	0101	000	WO	DBGDTRTXint	DBGDTRTXint
1110	000	0000	0110	000	RW	DBGWFAR	DBGWFAR
1110	000	0000	0110	010	RW	DBGOSECCR	DBGOSECCR
1110	000	0000	0111	000	RW	DBGVCR	DBGVCR
1110	000	0000	m[3:0]	100	RW	DBGBVR<m>	DBGBVR[]
1110	000	0000	m[3:0]	101	RW	DBGBCR<m>	DBGBCR[]
1110	000	0000	m[3:0]	110	RW	DBGWVR<m>	DBGWVR[]
1110	000	0000	m[3:0]	111	RW	DBGWCR<m>	DBGWCR[]
1110	000	0001	0000	000	RO	DBGDRAR	DBGDRAR
1110	000	0001	0000	100	WO	DBGOSLAR	DBGOSLAR
1110	000	0001	0001	100	RO	DBGOSLSR	DBGOSLSR
1110	000	0001	0011	100	RW	DBGOSDLR	DBGOSDLR
1110	000	0001	0100	100	RW	DBGPRCR	DBGPRCR
1110	000	0001	m[3:0]	001	RW	DBGBXVR<m>	DBGBXVR[]
1110	000	0010	0000	000	RO	DBGDSAR	DBGDSAR
1110	000	0111	0000	111	RO	DBGDEVID2	DBGDEVID2

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1110	000	0111	0001	111	RO	DBGDEVID1	DBGDEVID1
1110	000	0111	0010	111	RO	DBGDEVID	DBGDEVID
1110	000	0111	1000	110	RW	DBGCLAIMSET	DBGCLAIMSET
1110	000	0111	1001	110	RW	DBGCLAIMCLR	DBGCLAIMCLR
1110	000	0111	1110	110	RO	DBGAUTHSTATUS	DBGAUTHSTATUS
1110	111	0000	0000	000	RO	JIDR	JIDR
1110	111	0001	0000	000	RO	JOSCR	JOSCR
1110	111	0010	0000	000	RO	JMCR	JMCR
1111	000	0000	0000	000	RO	MIDR	MIDR
1111	000	0000	0000	000	RO	MIDR	VPIDR
1111	000	0000	0000	000	RO	MIDR	VPIDR_EL2
1111	000	0000	0000	000	RO	MIDR	MIDR
1111	000	0000	0000	000	RO	MIDR	VPIDR
1111	000	0000	0000	000	RO	MIDR	VPIDR_EL2
1111	000	0000	0000	001	RO	CTR	CTR
1111	000	0000	0000	010	RO	TCMTR	TCMTR
1111	000	0000	0000	011	RO	TLBTR	TLBTR
1111	000	0000	0000	101	RO	MPIDR	MPIDR
1111	000	0000	0000	101	RO	MPIDR	VMPIDR
1111	000	0000	0000	101	RO	MPIDR	VMPIDR_EL2
1111	000	0000	0000	101	RO	MPIDR	MPIDR
1111	000	0000	0000	101	RO	MPIDR	VMPIDR
1111	000	0000	0000	101	RO	MPIDR	VMPIDR_EL2
1111	000	0000	0000	110	RO	REVIDR	REVIDR
1111	000	0000	0001	000	RO	ID_PFR0	ID_PFR0
1111	000	0000	0001	001	RO	ID_PFR1	ID_PFR1
1111	000	0000	0001	010	RO	ID_DFR0	ID_DFR0
1111	000	0000	0001	011	RO	ID_AFR0	ID_AFR0
1111	000	0000	0001	100	RO	ID_MMFR0	ID_MMFR0
1111	000	0000	0001	101	RO	ID_MMFR1	ID_MMFR1
1111	000	0000	0001	110	RO	ID_MMFR2	ID_MMFR2
1111	000	0000	0001	111	RO	ID_MMFR3	ID_MMFR3
1111	000	0000	0010	000	RO	ID_ISAR0	ID_ISAR0
1111	000	0000	0010	001	RO	ID_ISAR1	ID_ISAR1
1111	000	0000	0010	010	RO	ID_ISAR2	ID_ISAR2
1111	000	0000	0010	011	RO	ID_ISAR3	ID_ISAR3
1111	000	0000	0010	100	RO	ID_ISAR4	ID_ISAR4
1111	000	0000	0010	101	RO	ID_ISAR5	ID_ISAR5
1111	000	0000	0010	110	RO	ID_MMFR4	ID_MMFR4
1111	000	0000	0010	111	RO	ID_ISAR6	ID_ISAR6
1111	000	0000	0011	100	RO	ID_PFR2	ID_PFR2
1111	000	0000	0011	101	RO	ID_DFR1	ID_DFR1
1111	000	0000	0011	110	RO	ID_MMFR5	ID_MMFR5
1111	000	0001	0000	000	RW	SCTLR	SCTLR
1111	000	0001	0000	000	RW	SCTLR	SCTLR_NS
1111	000	0001	0000	000	RW	SCTLR	SCTLR_S
1111	000	0001	0000	001	RW	ACTLR	ACTLR
1111	000	0001	0000	001	RW	ACTLR	ACTLR_NS
1111	000	0001	0000	001	RW	ACTLR	ACTLR_S

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	0001	0000	010	RW	CPACR	CPACR
1111	000	0001	0000	011	RW	ACTLR2	ACTLR2
1111	000	0001	0000	011	RW	ACTLR2	ACTLR2_NS
1111	000	0001	0000	011	RW	ACTLR2	ACTLR2_S
1111	000	0001	0001	000	RW	SCR	SCR
1111	000	0001	0001	001	RW	SDER	SDER
1111	000	0001	0001	010	RW	NSACR	NSACR
1111	000	0001	0010	001	RW	TRFCR	TRFCR
1111	000	0001	0011	001	RW	SDCR	SDCR
1111	000	0010	0000	000	RO	TTBR0	TTBR0
1111	000	0010	0000	000	RO	TTBR0	TTBR0_NS
1111	000	0010	0000	000	RO	TTBR0	TTBR0_S
1111	000	0010	0000	001	RO	TTBR1	TTBR1
1111	000	0010	0000	001	RO	TTBR1	TTBR1_NS
1111	000	0010	0000	001	RO	TTBR1	TTBR1_S
1111	000	0010	0000	010	RW	TTBCR	TTBCR
1111	000	0010	0000	010	RW	TTBCR	TTBCR_NS
1111	000	0010	0000	010	RW	TTBCR	TTBCR_S
1111	000	0010	0000	011	RW	TTBCR2	TTBCR2
1111	000	0010	0000	011	RW	TTBCR2	TTBCR2_NS
1111	000	0010	0000	011	RW	TTBCR2	TTBCR2_S
1111	000	0011	0000	000	RW	DACR	DACR
1111	000	0011	0000	000	RW	DACR	DACR_NS
1111	000	0011	0000	000	RW	DACR	DACR_S
1111	000	0100	0110	000	RW	ICC_PMR	ICC_PMR
1111	000	0100	0110	000	RW	ICC_PMR	ICV_PMR
1111	000	0100	0110	000	RW	ICC_PMR	ICC_PMR
1111	000	0100	0110	000	RW	ICC_PMR	ICV_PMR
1111	000	0101	0000	000	RW	DFSR	DFSR
1111	000	0101	0000	000	RW	DFSR	DFSR_NS
1111	000	0101	0000	000	RW	DFSR	DFSR_S
1111	000	0101	0000	001	RW	IFSR	IFSR
1111	000	0101	0000	001	RW	IFSR	IFSR_NS
1111	000	0101	0000	001	RW	IFSR	IFSR_S
1111	000	0101	0001	000	RW	ADFSR	ADFSR
1111	000	0101	0001	000	RW	ADFSR	ADFSR_NS
1111	000	0101	0001	000	RW	ADFSR	ADFSR_S
1111	000	0101	0001	001	RW	AIFSR	AIFSR
1111	000	0101	0001	001	RW	AIFSR	AIFSR_NS
1111	000	0101	0001	001	RW	AIFSR	AIFSR_S
1111	000	0101	0011	000	RO	ERRIDR	ERRIDR
1111	000	0101	0011	001	RW	ERRSEL	ERRSEL
1111	000	0101	0100	000	RO	ERXFR	ERXFR
1111	000	0101	0100	001	RW	ERXCTLR	ERXCTLR
1111	000	0101	0100	010	RW	ERXSTATUS	ERXSTATUS
1111	000	0101	0100	011	RW	ERXADDR	ERXADDR
1111	000	0101	0100	100	RO	ERXFR2	ERXFR2
1111	000	0101	0100	101	RW	ERXCTLR2	ERXCTLR2
1111	000	0101	0100	111	RW	ERXADDR2	ERXADDR2

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	0101	0101	000	RW	ERXMISC0	ERXMISC0
1111	000	0101	0101	001	RW	ERXMISC1	ERXMISC1
1111	000	0101	0101	010	RW	ERXMISC4	ERXMISC4
1111	000	0101	0101	011	RW	ERXMISC5	ERXMISC5
1111	000	0101	0101	100	RW	ERXMISC2	ERXMISC2
1111	000	0101	0101	101	RW	ERXMISC3	ERXMISC3
1111	000	0101	0101	110	RW	ERXMISC6	ERXMISC6
1111	000	0101	0101	111	RW	ERXMISC7	ERXMISC7
1111	000	0110	0000	000	RW	DFAR	DFAR
1111	000	0110	0000	000	RW	DFAR	DFAR_NS
1111	000	0110	0000	000	RW	DFAR	DFAR_S
1111	000	0110	0000	010	RW	IFAR	IFAR
1111	000	0110	0000	010	RW	IFAR	IFAR_NS
1111	000	0110	0000	010	RW	IFAR	IFAR_S
1111	000	0111	0001	000	-	ICIALLUIS	-
1111	000	0111	0001	110	-	BPIALLIS	-
1111	000	0111	0011	100	-	CFPRCTX	-
1111	000	0111	0011	101	-	DVPRCTX	-
1111	000	0111	0011	110	-	COSPRCTX	-
1111	000	0111	0011	111	-	CPPRCTX	-
1111	000	0111	0100	000	RO	PAR	PAR
1111	000	0111	0100	000	RO	PAR	PAR_NS
1111	000	0111	0100	000	RO	PAR	PAR_S
1111	000	0111	0101	000	-	ICIALLU	-
1111	000	0111	0101	001	-	ICIMVAU	-
1111	000	0111	0101	100	-	CP15ISB	-
1111	000	0111	0101	110	-	BPIALL	-
1111	000	0111	0101	111	-	BPIMVA	-
1111	000	0111	0110	001	-	DCIMVAC	-
1111	000	0111	0110	010	-	DCISW	-
1111	000	0111	1000	000	-	ATS1CPR	-
1111	000	0111	1000	001	-	ATS1CPW	-
1111	000	0111	1000	010	-	ATS1CUR	-
1111	000	0111	1000	011	-	ATS1CUW	-
1111	000	0111	1000	100	-	ATS12NSOPR	-
1111	000	0111	1000	101	-	ATS12NSOPW	-
1111	000	0111	1000	110	-	ATS12NSOUR	-
1111	000	0111	1000	111	-	ATS12NSOUW	-
1111	000	0111	1001	000	-	ATS1CPRP	-
1111	000	0111	1001	001	-	ATS1CPWP	-
1111	000	0111	1010	001	-	DCCMVAC	-
1111	000	0111	1010	010	-	DCCSW	-
1111	000	0111	1010	100	-	CP15DSB	-
1111	000	0111	1010	101	-	CP15DMB	-
1111	000	0111	1011	001	-	DCCMVAU	-
1111	000	0111	1110	001	-	DCCIMVAC	-
1111	000	0111	1110	010	-	DCCISW	-
1111	000	1000	0011	000	-	TLBIALLIS	-
1111	000	1000	0011	001	-	TLBIMVAIS	-

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	1000	0011	010	-	TLBIASIDIS	-
1111	000	1000	0011	011	-	TLBIMVA AIS	-
1111	000	1000	0011	101	-	TLBIMVALIS	-
1111	000	1000	0011	111	-	TLBIMVAALIS	-
1111	000	1000	0101	000	-	ITLBIALL	-
1111	000	1000	0101	001	-	ITLBIMVA	-
1111	000	1000	0101	010	-	ITLBIASID	-
1111	000	1000	0110	000	-	DTLBIALL	-
1111	000	1000	0110	001	-	DTLBIMVA	-
1111	000	1000	0110	010	-	DTLBIASID	-
1111	000	1000	0111	000	-	TLBIALL	-
1111	000	1000	0111	001	-	TLBIMVA	-
1111	000	1000	0111	010	-	TLBIASID	-
1111	000	1000	0111	011	-	TLBIMVAA	-
1111	000	1000	0111	101	-	TLBIMVAL	-
1111	000	1000	0111	111	-	TLBIMVAAL	-
1111	000	1001	1100	000	RW	PMCR	PMCR
1111	000	1001	1100	001	RW	PMCNTENSET	PMCNTENSET
1111	000	1001	1100	010	RW	PMCNTENCLR	PMCNTENCLR
1111	000	1001	1100	011	RW	PMOVS R	PMOVS R
1111	000	1001	1100	100	WO	PMSWINC	PMSWINC
1111	000	1001	1100	101	RW	PMSEL R	PMSEL R
1111	000	1001	1100	110	RO	PMCEID0	PMCEID0
1111	000	1001	1100	111	RO	PMCEID1	PMCEID1
1111	000	1001	1101	000	RO	PMCCNTR	PMCCNTR
1111	000	1001	1101	001	RW	PMXEVTYPER	PMCCFILTR
1111	000	1001	1101	001	RW	PMXEVTYPER	PMEVTYPER
1111	000	1001	1101	010	RW	PMXVCNTR	PMEVCNTR
1111	000	1001	1110	000	RW	PMUSERENR	PMUSERENR
1111	000	1001	1110	001	RW	PMINTENSET	PMINTENSET
1111	000	1001	1110	010	RW	PMINTENCLR	PMINTENCLR
1111	000	1001	1110	011	RW	PMOVSSET	PMOVSSET
1111	000	1001	1110	100	RO	PMCEID2	PMCEID2
1111	000	1001	1110	101	RO	PMCEID3	PMCEID3
1111	000	1001	1110	110	RO	PMMIR	PMMIR
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0_NS
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0_S
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR_NS
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR_S
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0_NS
1111	000	1010	0010	000	RW	PRRR-MAIR0	MAIR0_S
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR_NS
1111	000	1010	0010	000	RW	PRRR-MAIR0	PRRR_S
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1_NS

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1_S
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR_NS
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR_S
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1_NS
1111	000	1010	0010	001	RW	NMRR-MAIR1	MAIR1_S
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR_NS
1111	000	1010	0010	001	RW	NMRR-MAIR1	NMRR_S
1111	000	1010	0011	000	RW	AMAIRO	AMAIRO
1111	000	1010	0011	000	RW	AMAIRO	AMAIRO_NS
1111	000	1010	0011	000	RW	AMAIRO	AMAIRO_S
1111	000	1010	0011	001	RW	AMAIR1	AMAIR1
1111	000	1010	0011	001	RW	AMAIR1	AMAIR1_NS
1111	000	1010	0011	001	RW	AMAIR1	AMAIR1_S
1111	000	1100	0000	000	RW	VBAR	VBAR
1111	000	1100	0000	000	RW	VBAR	VBAR_NS
1111	000	1100	0000	000	RW	VBAR	VBAR_S
1111	000	1100	0000	001	RO	RVBAR-MVBAR	RVBAR
1111	000	1100	0000	001	RO	RVBAR-MVBAR	MVBAR
1111	000	1100	0000	001	RO	RVBAR-MVBAR	RVBAR
1111	000	1100	0000	001	RW	RVBAR-MVBAR	MVBAR
1111	000	1100	0000	010	RW	RMR	RMR
1111	000	1100	0001	000	RO	ISR	ISR
1111	000	1100	0001	001	RW	DISR	DISR
1111	000	1100	0001	001	RW	DISR	VDISR
1111	000	1100	0001	001	RW	DISR	VDISR_EL2
1111	000	1100	0001	001	RW	DISR	DISR
1111	000	1100	0001	001	RW	DISR	VDISR
1111	000	1100	0001	001	RW	DISR	VDISR_EL2
1111	000	1100	1000	000	RO	ICC_IAR0	ICC_IAR0
1111	000	1100	1000	000	RO	ICC_IAR0	ICV_IAR0
1111	000	1100	1000	000	RO	ICC_IAR0	ICC_IAR0
1111	000	1100	1000	000	RO	ICC_IAR0	ICV_IAR0
1111	000	1100	1000	001	WO	ICC_EOIR0	ICC_EOIR0
1111	000	1100	1000	001	WO	ICC_EOIR0	ICV_EOIR0
1111	000	1100	1000	001	WO	ICC_EOIR0	ICC_EOIR0
1111	000	1100	1000	001	WO	ICC_EOIR0	ICV_EOIR0
1111	000	1100	1000	010	RO	ICC_HPPIR0	ICC_HPPIR0
1111	000	1100	1000	010	RO	ICC_HPPIR0	ICV_HPPIR0
1111	000	1100	1000	010	RO	ICC_HPPIR0	ICC_HPPIR0
1111	000	1100	1000	010	RO	ICC_HPPIR0	ICV_HPPIR0
1111	000	1100	1000	011	RW	ICC_BPR0	ICC_BPR0
1111	000	1100	1000	011	RW	ICC_BPR0	ICV_BPR0
1111	000	1100	1000	011	RW	ICC_BPR0	ICC_BPR0
1111	000	1100	1000	011	RW	ICC_BPR0	ICV_BPR0
1111	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>	ICC_AP0R[]
1111	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>	ICV_AP0R[]

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>	ICC_AP0R[]
1111	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>	ICV_AP0R[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R_NS[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R_S[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICV_AP1R[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R_NS[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICC_AP1R_S[]
1111	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>	ICV_AP1R[]
1111	000	1100	1011	001	WO	ICC_DIR	ICC_DIR
1111	000	1100	1011	001	WO	ICC_DIR	ICV_DIR
1111	000	1100	1011	001	WO	ICC_DIR	ICC_DIR
1111	000	1100	1011	001	WO	ICC_DIR	ICV_DIR
1111	000	1100	1011	011	RO	ICC_RPR	ICC_RPR
1111	000	1100	1011	011	RO	ICC_RPR	ICV_RPR
1111	000	1100	1011	011	RO	ICC_RPR	ICC_RPR
1111	000	1100	1011	011	RO	ICC_RPR	ICV_RPR
1111	000	1100	1100	000	RO	ICC_IAR1	ICC_IAR1
1111	000	1100	1100	000	RO	ICC_IAR1	ICV_IAR1
1111	000	1100	1100	000	RO	ICC_IAR1	ICC_IAR1
1111	000	1100	1100	000	RO	ICC_IAR1	ICV_IAR1
1111	000	1100	1100	001	WO	ICC_EOIR1	ICC_EOIR1
1111	000	1100	1100	001	WO	ICC_EOIR1	ICV_EOIR1
1111	000	1100	1100	001	WO	ICC_EOIR1	ICC_EOIR1
1111	000	1100	1100	001	WO	ICC_EOIR1	ICV_EOIR1
1111	000	1100	1100	010	RO	ICC_HPPIR1	ICC_HPPIR1
1111	000	1100	1100	010	RO	ICC_HPPIR1	ICV_HPPIR1
1111	000	1100	1100	010	RO	ICC_HPPIR1	ICC_HPPIR1
1111	000	1100	1100	010	RO	ICC_HPPIR1	ICV_HPPIR1
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1_NS
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1_S
1111	000	1100	1100	011	RW	ICC_BPR1	ICV_BPR1
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1_NS
1111	000	1100	1100	011	RW	ICC_BPR1	ICC_BPR1_S
1111	000	1100	1100	011	RW	ICC_BPR1	ICV_BPR1
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR_NS
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR_S
1111	000	1100	1100	100	RW	ICC_CTLR	ICV_CTLR
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR_NS
1111	000	1100	1100	100	RW	ICC_CTLR	ICC_CTLR_S
1111	000	1100	1100	100	RW	ICC_CTLR	ICV_CTLR
1111	000	1100	1100	101	RW	ICC_SRE	ICC_SRE
1111	000	1100	1100	101	RW	ICC_SRE	ICC_SRE_NS
1111	000	1100	1100	101	RW	ICC_SRE	ICC_SRE_S

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	1100	1100	110	RW	ICC_IGRPEN0	ICC_IGRPEN0
1111	000	1100	1100	110	RW	ICC_IGRPEN0	ICV_IGRPEN0
1111	000	1100	1100	110	RW	ICC_IGRPEN0	ICC_IGRPEN0
1111	000	1100	1100	110	RW	ICC_IGRPEN0	ICV_IGRPEN0
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1_NS
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1_S
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICV_IGRPEN1
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1_NS
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICC_IGRPEN1_S
1111	000	1100	1100	111	RW	ICC_IGRPEN1	ICV_IGRPEN1
1111	000	1101	0000	000	RW	FCSEIDR	FCSEIDR
1111	000	1101	0000	001	RW	CONTEXTIDR	CONTEXTIDR
1111	000	1101	0000	001	RW	CONTEXTIDR	CONTEXTIDR_NS
1111	000	1101	0000	001	RW	CONTEXTIDR	CONTEXTIDR_S
1111	000	1101	0000	010	RW	TPIDRURW	TPIDRURW
1111	000	1101	0000	010	RW	TPIDRURW	TPIDRURW_NS
1111	000	1101	0000	010	RW	TPIDRURW	TPIDRURW_S
1111	000	1101	0000	011	RW	TPIDRURO	TPIDRURO
1111	000	1101	0000	011	RW	TPIDRURO	TPIDRURO_NS
1111	000	1101	0000	011	RW	TPIDRURO	TPIDRURO_S
1111	000	1101	0000	100	RW	TPIDRPRW	TPIDRPRW
1111	000	1101	0000	100	RW	TPIDRPRW	TPIDRPRW_NS
1111	000	1101	0000	100	RW	TPIDRPRW	TPIDRPRW_S
1111	000	1101	0010	000	RW	AMCR	AMCR
1111	000	1101	0010	001	RO	AMCFGR	AMCFGR
1111	000	1101	0010	010	RO	AMCGCR	AMCGCR
1111	000	1101	0010	011	RW	AMUSERENR	AMUSERENR
1111	000	1101	0010	100	RW	AMCNTENCLR0	AMCNTENCLR0
1111	000	1101	0010	101	RW	AMCNTENSET0	AMCNTENSET0
1111	000	1101	0011	000	RW	AMCNTENCLR1	AMCNTENCLR1
1111	000	1101	0011	001	RW	AMCNTENSET1	AMCNTENSET1
1111	000	1101	011:m[3]	m[2:0]	RO	AMEVTYPEP0<m>	AMEVTYPEP0[]
1111	000	1101	111:m[3]	m[2:0]	RW	AMEVTYPEP1<m>	AMEVTYPEP1[]
1111	000	1110	0000	000	RW	CNTFRQ	CNTFRQ
1111	000	1110	0001	000	RW	CNTKCTL	CNTKCTL
1111	000	1110	0010	000	-	CNTP_TVAL	-
1111	000	1110	0010	000	-	CNTP_TVAL	-
1111	000	1110	0010	000	-	CNTP_TVAL	-
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHPS_CTL_EL1
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHP_CTL_EL2
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_NS
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_S
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHPS_CTL_EL1
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHP_CTL_EL2
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_NS

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_S
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHPS_CTL_EL1
1111	000	1110	0010	001	RW	CNTP_CTL	CNTHP_CTL_EL2
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_NS
1111	000	1110	0010	001	RW	CNTP_CTL	CNTP_CTL_S
1111	000	1110	0011	000	-	CNTV_TVAL	-
1111	000	1110	0011	000	-	CNTV_TVAL	-
1111	000	1110	0011	000	-	CNTV_TVAL	-
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHVS_CTL_EL1
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHV_CTL_EL2
1111	000	1110	0011	001	RW	CNTV_CTL	CNTV_CTL
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHVS_CTL_EL1
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHV_CTL_EL2
1111	000	1110	0011	001	RW	CNTV_CTL	CNTV_CTL
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHVS_CTL_EL1
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHV_CTL_EL2
1111	000	1110	0011	001	RW	CNTV_CTL	CNTV_CTL
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHVS_CTL_EL1
1111	000	1110	0011	001	RW	CNTV_CTL	CNTHV_CTL_EL2
1111	000	1110	0011	001	RW	CNTV_CTL	CNTV_CTL
1111	000	1110	10:m[4:3]	m[2:0]	RW	PMEVCNTR<m>	PMEVCNTR[]
1111	000	1110	1111	111	RW	PMCCFILTR	PMCCFILTR
1111	000	1110	11:m[4:3]	m[2:0]	RW	PMEVTYPEPER<m>	PMEVTYPEPER[]
1111	001	0000	0000	000	RO	CCSIDR	CCSIDR
1111	001	0000	0000	001	RO	CLIDR	CLIDR
1111	001	0000	0000	010	RO	CCSIDR2	CCSIDR2
1111	001	0000	0000	111	RO	AIDR	AIDR
1111	010	0000	0000	000	RW	CSSELR	CSSELR
1111	010	0000	0000	000	RW	CSSELR	CSSELR_NS
1111	010	0000	0000	000	RW	CSSELR	CSSELR_S
1111	011	0100	0101	000	RW	DSPSR	DSPSR
1111	011	0100	0101	001	RW	DLR	DLR
1111	011	0100	0101	010	RW	DSPSR2	DSPSR2
1111	100	0000	0000	000	RO	VPIDR	MIDR
1111	100	0000	0000	000	RW	VPIDR	VPIDR
1111	100	0000	0000	101	RO	VMPIDR	MPIDR
1111	100	0000	0000	101	RW	VMPIDR	VMPIDR
1111	100	0001	0000	000	RW	HSCTLR	HSCTLR
1111	100	0001	0000	001	RW	HACTLR	HACTLR
1111	100	0001	0000	011	RW	HACTLR2	HACTLR2
1111	100	0001	0001	000	RW	HCR	HCR
1111	100	0001	0001	001	RW	HDCR	HDCR
1111	100	0001	0001	010	RW	HCPTR	HCPTR
1111	100	0001	0001	011	RW	HSTR	HSTR
1111	100	0001	0001	100	RW	HCR2	HCR2
1111	100	0001	0001	111	RW	HACR	HACR
1111	100	0001	0010	001	RW	HTRFCR	HTRFCR
1111	100	0010	0000	010	RW	HTCR	HTCR
1111	100	0010	0001	010	RW	VTCR	VTCR
1111	100	0101	0001	000	RW	HADFSR	HADFSR
1111	100	0101	0001	001	RW	HAIFSR	HAIFSR

coproc	opc1	CRn	CRm	opc2	Access	Mnemonic	Register
1111	100	0101	0010	000	RW	HSR	HSR
1111	100	0101	0010	011	RW	VDFSR	VDFSR
1111	100	0110	0000	000	RW	HDFAR	HDFAR
1111	100	0110	0000	010	RW	HIFAR	HIFAR
1111	100	0110	0000	100	RW	HPFAR	HPFAR
1111	100	0111	1000	000	-	ATS1HR	-
1111	100	0111	1000	001	-	ATS1HW	-
1111	100	1000	0000	001	-	TLBIIPAS2IS	-
1111	100	1000	0000	101	-	TLBIIPAS2LIS	-
1111	100	1000	0011	000	-	TLBIALLHIS	-
1111	100	1000	0011	001	-	TLBIMVAHIS	-
1111	100	1000	0011	100	-	TLBIALLNSNHIS	-
1111	100	1000	0011	101	-	TLBIMVALHIS	-
1111	100	1000	0100	001	-	TLBIIPAS2	-
1111	100	1000	0100	101	-	TLBIIPAS2L	-
1111	100	1000	0111	000	-	TLBIALLH	-
1111	100	1000	0111	001	-	TLBIMVAH	-
1111	100	1000	0111	100	-	TLBIALLNSNH	-
1111	100	1000	0111	101	-	TLBIMVALH	-
1111	100	1010	0010	000	RW	HMAIR0	HMAIR0
1111	100	1010	0010	001	RW	HMAIR1	HMAIR1
1111	100	1010	0011	000	RW	HAMAIRO	HAMAIRO
1111	100	1010	0011	001	RW	HAMAIR1	HAMAIR1
1111	100	1100	0000	000	RW	HVBAR	HVBAR
1111	100	1100	0000	010	RW	HRMR	HRMR
1111	100	1100	0001	001	RW	VDISR	VDISR
1111	100	1100	1000	0:m[1:0]	RW	ICH_AP0R<m>	ICH_AP0R[]
1111	100	1100	1001	0:m[1:0]	RW	ICH_AP1R<m>	ICH_AP1R[]
1111	100	1100	1001	101	RW	ICC_HSRE	ICC_HSRE
1111	100	1100	1011	000	RW	ICH_HCR	ICH_HCR
1111	100	1100	1011	001	RO	ICH_VTR	ICH_VTR
1111	100	1100	1011	010	RO	ICH_MISR	ICH_MISR
1111	100	1100	1011	011	RO	ICH_EISR	ICH_EISR
1111	100	1100	1011	101	RO	ICH_ELRSR	ICH_ELRSR
1111	100	1100	1011	111	RW	ICH_VMCR	ICH_VMCR
1111	100	1100	110:m[3]	m[2:0]	RW	ICH_LR<m>	ICH_LR[]
1111	100	1100	111:m[3]	m[2:0]	RW	ICH_LRC<m>	ICH_LRC[]
1111	100	1101	0000	010	RW	HTPIDR	HTPIDR
1111	100	1110	0001	000	RW	CNTHCTL	CNTHCTL
1111	100	1110	0010	000	-	CNTHP_TVAL	-
1111	100	1110	0010	001	RW	CNTHP_CTL	CNTHP_CTL
1111	110	1100	1100	100	RW	ICC_MCTLR	ICC_MCTLR
1111	110	1100	1100	101	RW	ICC_MSRE	ICC_MSRE
1111	110	1100	1100	111	RW	ICC_MGRPEN1	ICC_MGRPEN1

Accessed using MCRR/MRRC:

coproc	opc1	CRm	Access	Mnemonic	Register
1110	0000	0001	-	DBGDRAR	-

coproc	opc1	CRm	Access	Mnemonic	Register
1110	0000	0010	-	DBGDSAR	-
1111	0000	0010	WO	TTBR0	TTBR0
1111	0000	0010	WO	TTBR0	TTBR0_NS
1111	0000	0010	WO	TTBR0	TTBR0_S
1111	0000	0111	WO	PAR	PAR
1111	0000	0111	WO	PAR	PAR_NS
1111	0000	0111	WO	PAR	PAR_S
1111	0000	1001	WO	PMCCNTR	PMCCNTR
1111	0000	1100	WO	ICC_SGI1R	ICC_SGI1R
1111	0000	1110	-	CNTPCT	-
1111	0001	0010	WO	TTBR1	TTBR1
1111	0001	0010	WO	TTBR1	TTBR1_NS
1111	0001	0010	WO	TTBR1	TTBR1_S
1111	0001	1100	WO	ICC_ASgi1R	ICC_ASgi1R
1111	0001	1110	-	CNTVCT	-
1111	0010	1100	WO	ICC_SGI0R	ICC_SGI0R
1111	0010	1110	WO	CNTP_CVAL	CNTHPS_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTHP_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_NS
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_S
1111	0010	1110	WO	CNTP_CVAL	CNTHPS_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTHP_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_NS
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_S
1111	0010	1110	WO	CNTP_CVAL	CNTHPS_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTHP_CVAL_EL2
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_NS
1111	0010	1110	WO	CNTP_CVAL	CNTP_CVAL_S
1111	0011	1110	WO	CNTV_CVAL	CNTHVS_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTHV_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTV_CVAL
1111	0011	1110	WO	CNTV_CVAL	CNTHVS_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTHV_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTV_CVAL
1111	0011	1110	WO	CNTV_CVAL	CNTHVS_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTHV_CVAL_EL2
1111	0011	1110	WO	CNTV_CVAL	CNTV_CVAL
1111	0100	0010	WO	HTTBR	HTTBR
1111	0100	1110	WO	CNTVOFF	CNTVOFF
1111	0110	0010	WO	VTTBR	VTTBR
1111	0110	1110	WO	CNTHP_CVAL	CNTHP_CVAL
1111	0:m[2:0]	000:m[3]	WO	AMEVCNTR0<m>	AMEVCNTR0[]
1111	0:m[2:0]	010:m[3]	WO	AMEVCNTR1<m>	AMEVCNTR1[]
1111	1000	1110	-	CNTPCTSS	-
1111	1001	1110	-	CNTVCTSS	-

Accessed using MRS/MSR:

R	M	M1	Mnemonic
0	1	1110	ELR_hyp
1	0	1110	SPSR_fiq
1	1	0000	SPSR_irq
1	1	0010	SPSR_svc
1	1	0100	SPSR_abt
1	1	0110	SPSR_und
1	1	1100	SPSR_mon
1	1	1110	SPSR_hyp

Accessed using VMRS/VMSR:

reg	Access	Mnemonic	Register
0000	RO	FPSID	FPSID
0001	RW	FPSCR	FPSCR
0101	RO	MVFR2	MVFR2
0110	RO	MVFR1	MVFR1
0111	RO	MVFR0	MVFR0
1000	RW	FPEXC	FPEXC

Registers and operations in AArch64

Accessed using AT:

op0	op1	CRn	CRm	op2	Mnemonic
01	000	0111	1000	000	AT S1E1R
01	000	0111	1000	001	AT S1E1W
01	000	0111	1000	010	AT S1E0R
01	000	0111	1000	011	AT S1E0W
01	000	0111	1001	000	AT S1E1RP
01	000	0111	1001	001	AT S1E1WP
01	100	0111	1000	000	AT S1E2R
01	100	0111	1000	001	AT S1E2W
01	100	0111	1000	100	AT S12E1R
01	100	0111	1000	101	AT S12E1W
01	100	0111	1000	110	AT S12E0R
01	100	0111	1000	111	AT S12E0W
01	110	0111	1000	000	AT S1E3R
01	110	0111	1000	001	AT S1E3W

Accessed using BRB:

op0	op1	CRn	CRm	op2	Mnemonic
01	001	0111	0010	100	BRB IALL
01	001	0111	0010	101	BRB INJ

Accessed using CFP:

op0	op1	CRn	CRm	op2	Mnemonic
01	011	0111	0011	100	CFP_RCTX

Accessed using CPP:

op0	op1	CRn	CRm	op2	Mnemonic
01	011	0111	0011	111	CPP_RCTX

Accessed using DC:

op0	op1	CRn	CRm	op2	Mnemonic
01	000	0111	0110	001	DC_IVAC
01	000	0111	0110	010	DC_ISW
01	000	0111	0110	011	DC_IGVAC
01	000	0111	0110	100	DC_IGSW
01	000	0111	0110	101	DC_IGDVAC
01	000	0111	0110	110	DC_IGDSW
01	000	0111	1010	010	DC_CSW
01	000	0111	1010	100	DC_CGSW
01	000	0111	1010	110	DC_CGDSW
01	000	0111	1110	010	DC_CISW
01	000	0111	1110	100	DC_CIGSW
01	000	0111	1110	110	DC_CIGDSW
01	011	0111	0010	111	DC_TRCIT
01	011	0111	0100	001	DC_ZVA
01	011	0111	0100	011	DC_GVA
01	011	0111	0100	100	DC_GZVA
01	011	0111	1010	001	DC_CVAC
01	011	0111	1010	011	DC_CGVAC
01	011	0111	1010	101	DC_CGDVAC
01	011	0111	1011	001	DC_CVAU
01	011	0111	1100	001	DC_CVAP
01	011	0111	1100	011	DC_CGVAP
01	011	0111	1100	101	DC_CGDVAP
01	011	0111	1101	001	DC_CVADP
01	011	0111	1101	011	DC_CGVADP
01	011	0111	1101	101	DC_CGDVADP
01	011	0111	1110	001	DC_CIVAC
01	011	0111	1110	011	DC_CIGVAC
01	011	0111	1110	101	DC_CIGDVAC
01	100	0111	1110	000	DC_CIPAE
01	100	0111	1110	111	DC_CIGDPAE
01	110	0111	1110	001	DC_CIPAPA
01	110	0111	1110	101	DC_CIGDPAPA
01	110	0111	1111	001	DC_CIPAE
01	110	0111	1111	101	DC_CIGDPAE

Accessed using DVP:

op0	op1	CRn	CRm	op2	Mnemonic
01	011	0111	0011	101	DVP RCTX

Accessed using IC:

op0	op1	CRn	CRm	op2	Mnemonic
01	000	0111	0001	000	IC IALLUIS
01	000	0111	0101	000	IC IALLU
01	011	0111	0101	001	IC IVAU

Accessed using MRS/MSR:

op0	op1	CRn	CRm	op2	Access	Mnemonic
10	000	0000	0000	010	RW	OSDTRRX_EL1
10	000	0000	0010	000	RW	MDCCINT_EL1
10	000	0000	0010	010	RW	MDSCR_EL1
10	000	0000	0011	010	RW	OSDTRTX_EL1
10	000	0000	0100	010	RW	MDSELR_EL1
10	000	0000	0110	010	RW	OSECCR_EL1
10	000	0000	m[3:0]	100	RW	DBGBVR<m>_EL1
10	000	0000	m[3:0]	101	RW	DBGBCR<m>_EL1
10	000	0000	m[3:0]	110	RW	DBGWVR<m>_EL1
10	000	0000	m[3:0]	111	RW	DBGWCR<m>_EL1
10	000	0001	0000	000	RO	MDRAR_EL1
10	000	0001	0000	100	WO	OSLAR_EL1
10	000	0001	0001	100	RO	OSLSR_EL1
10	000	0001	0011	100	RW	OSDLR_EL1
10	000	0001	0100	100	RW	DBGPRCR_EL1
10	000	0111	1000	110	RW	DBGCLAIMSET_EL1
10	000	0111	1001	110	RW	DBGCLAIMCLR_EL1
10	000	0111	1110	110	RO	DBGAUTHSTATUS_EL1
10	000	1001	1101	00:n[0]	RO	SPMCGCR<n>_EL1
10	000	1001	1101	011	RW	SPMACCESSR_EL1
10	000	1001	1101	011	RW	SPMACCESSR_EL1
10	000	1001	1101	011	RW	SPMACCESSR_EL1
10	000	1001	1101	011	RW	SPMACCESSR_EL1
10	000	1001	1101	100	RO	SPMIIDR_EL1
10	000	1001	1101	101	RO	SPMDEVARCH_EL1
10	000	1001	1101	110	RO	SPMDEVAFF_EL1
10	000	1001	1101	111	RO	SPMCFGR_EL1
10	000	1001	1110	001	RW	SPMINTENSET_EL1
10	000	1001	1110	010	RW	SPMINTENCLR_EL1
10	000	1110	1011	111	RO	PMCCNTSVR_EL1
10	000	1110	10:m[4:3]	m[2:0]	RO	PMEVCNTSVR<m>_EL1
10	000	1110	1100	000	RO	PMICNTSVR_EL1
10	001	0000	0000	001	RW	TRCTRACEIDR
10	001	0000	0000	010	RW	TRCVICTLR
10	001	0000	0000	110	RO	TRCIDR8

op0	op1	CRn	CRm	op2	Access	Mnemonic
10	001	0000	0000	111	RW	TRCIMSPEC0
10	001	0000	0001	000	RW	TRCPRGCTLR
10	001	0000	0001	001	RW	TRCQCTLR
10	001	0000	0001	010	RW	TRCVIIECTLR
10	001	0000	0001	110	RO	TRCIDR9
10	001	0000	0010	001	RW	TRCITEEDCR
10	001	0000	0010	010	RW	TRCVISSCTLR
10	001	0000	0010	110	RO	TRCIDR10
10	001	0000	0011	000	RO	TRCSTATR
10	001	0000	0011	010	RW	TRCVIPCSSCTLR
10	001	0000	0011	110	RO	TRCIDR11
10	001	0000	00:m[1:0]	100	RW	TRCSEQEVR<m>
10	001	0000	00:m[1:0]	101	RW	TRCCNTRLDVR<m>
10	001	0000	0100	000	RW	TRCCONFIGR
10	001	0000	0100	110	RO	TRCIDR12
10	001	0000	0101	110	RO	TRCIDR13
10	001	0000	0110	000	RW	TRCAUXCTLR
10	001	0000	0110	100	RW	TRCSEQRSTEVR
10	001	0000	0111	100	RW	TRCSEQSTR
10	001	0000	01:m[1:0]	101	RW	TRCCNTCTLR<m>
10	001	0000	0:m[2:0]	111	RW	TRCIMSPEC<m>
10	001	0000	1000	000	RW	TRCEVENTCTL0R
10	001	0000	1000	111	RO	TRCIDR0
10	001	0000	1001	000	RW	TRCEVENTCTL1R
10	001	0000	1001	111	RO	TRCIDR1
10	001	0000	1010	000	RW	TRCRSR
10	001	0000	1010	111	RO	TRCIDR2
10	001	0000	1011	000	RW	TRCSTALLCTLR
10	001	0000	1011	111	RO	TRCIDR3
10	001	0000	10:m[1:0]	100	RW	TRCEXTINSELR<m>
10	001	0000	10:m[1:0]	101	RW	TRCCNTVR<m>
10	001	0000	1100	000	RW	TRCTSCTLR
10	001	0000	1100	111	RO	TRCIDR4
10	001	0000	1101	000	RW	TRCSYNCPR
10	001	0000	1101	111	RO	TRCIDR5
10	001	0000	1110	000	RW	TRCCCCTLR
10	001	0000	1110	111	RO	TRCIDR6
10	001	0000	1111	000	RW	TRCBBCTLR
10	001	0000	1111	111	RO	TRCIDR7
10	001	0001	0001	100	RO	TRCOSLSR
10	001	0001	0:m[2:0]	010	RW	TRCSSCCR<m>
10	001	0001	0:m[2:0]	011	RW	TRCSSPCICR<m>
10	001	0001	1:m[2:0]	010	RW	TRCSSCSR<m>
10	001	0001	m[3:0]	00:m[4]	RW	TRCRSCTLR<m>
10	001	0010	m[2:0]:0	00:m[3]	RW	TRCACVR<m>
10	001	0010	m[2:0]:0	01:m[3]	RW	TRCACATR<m>
10	001	0011	0000	010	RW	TRCCIDCCTLR0
10	001	0011	0001	010	RW	TRCCIDCCTLR1
10	001	0011	0010	010	RW	TRCVMIDCCTLR0

op0	op1	CRn	CRm	op2	Access	Mnemonic
10	001	0011	0011	010	RW	TRCVMIDCCTLR1
10	001	0011	m[2:0]:0	000	RW	TRCCIDCVR<m>
10	001	0011	m[2:0]:0	001	RW	TRCVMIDCVR<m>
10	001	0111	0010	111	RO	TRCDEVID
10	001	0111	1000	110	RW	TRCCLAIMSET
10	001	0111	1001	110	RW	TRCCLAIMCLR
10	001	0111	1110	110	RO	TRCAUTHSTATUS
10	001	0111	1111	110	RO	TRCDEVARCH
10	001	1000	m[3:0]	m[4]:00	RO	BRBINF<m>_EL1
10	001	1000	m[3:0]	m[4]:01	RO	BRBSRC<m>_EL1
10	001	1000	m[3:0]	m[4]:10	RO	BRBTGT<m>_EL1
10	001	1001	0000	000	RW	BRBCR_EL1
10	001	1001	0000	000	RW	BRBCR_EL1
10	001	1001	0000	000	RW	BRBCR_EL1
10	001	1001	0000	000	RW	BRBCR_EL1
10	001	1001	0000	001	RW	BRBFCR_EL1
10	001	1001	0000	010	RW	BRBTS_EL1
10	001	1001	0001	000	RW	BRBINFINJ_EL1
10	001	1001	0001	001	RW	BRBSRCINJ_EL1
10	001	1001	0001	010	RW	BRBTGTINJ_EL1
10	001	1001	0010	000	RO	BRBIDR0_EL1
10	011	0000	0001	000	RO	MDCCSR_EL0
10	011	0000	0100	000	RW	DBGDTR_EL0
10	011	0000	0101	000	RO	DBGDTRRX_EL0
10	011	0000	0101	000	WO	DBGDTRTX_EL0
10	011	1001	1100	000	RW	SPMCR_EL0
10	011	1001	1100	001	RW	SPMCNTENSET_EL0
10	011	1001	1100	010	RW	SPMCNTENCLR_EL0
10	011	1001	1100	011	RW	SPMOVSCLR_EL0
10	011	1001	1100	101	RW	SPMSELR_EL0
10	011	1001	1110	011	RW	SPMOVSSSET_EL0
10	011	1110	000:m[3]	m[2:0]	RW	SPMEVCNTR<m>_EL0
10	011	1110	001:m[3]	m[2:0]	RW	SPMEVTPER<m>_EL0
10	011	1110	010:m[3]	m[2:0]	RW	SPMEVFILTR<m>_EL0
10	011	1110	011:m[3]	m[2:0]	RW	SPMEVFILT2R<m>_EL0
10	100	0000	0111	000	RW	DBGVCR32_EL2
10	100	1001	0000	000	RW	BRBCR_EL2
10	100	1001	1101	011	RW	SPMACCESSR_EL2
10	101	1001	0000	000	RW	BRBCR_EL12
10	101	1001	1101	011	RW	SPMACCESSR_EL12
10	110	1001	1101	011	RW	SPMACCESSR_EL3
10	110	1001	1110	111	RW	SPMROOTCR_EL3
10	111	1001	1110	111	RW	SPMSCR_EL1
11	000	0000	0000	000	RO	MIDR_EL1
11	000	0000	0000	000	RO	MIDR_EL1
11	000	0000	0000	000	RO	MIDR_EL1
11	000	0000	0000	000	RO	MIDR_EL1
11	000	0000	0000	100	RO	MPUIR_EL1
11	000	0000	0000	101	RO	MPIDR_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	0000	0000	101	RO	MPIDR_EL1
11	000	0000	0000	101	RO	MPIDR_EL1
11	000	0000	0000	101	RO	MPIDR_EL1
11	000	0000	0000	110	RO	REVIDR_EL1
11	000	0000	0001	000	RO	ID_PFR0_EL1
11	000	0000	0001	001	RO	ID_PFR1_EL1
11	000	0000	0001	010	RO	ID_DFR0_EL1
11	000	0000	0001	011	RO	ID_AFR0_EL1
11	000	0000	0001	100	RO	ID_MMFR0_EL1
11	000	0000	0001	101	RO	ID_MMFR1_EL1
11	000	0000	0001	110	RO	ID_MMFR2_EL1
11	000	0000	0001	111	RO	ID_MMFR3_EL1
11	000	0000	0010	000	RO	ID_ISAR0_EL1
11	000	0000	0010	001	RO	ID_ISAR1_EL1
11	000	0000	0010	010	RO	ID_ISAR2_EL1
11	000	0000	0010	011	RO	ID_ISAR3_EL1
11	000	0000	0010	100	RO	ID_ISAR4_EL1
11	000	0000	0010	101	RO	ID_ISAR5_EL1
11	000	0000	0010	110	RO	ID_MMFR4_EL1
11	000	0000	0010	111	RO	ID_ISAR6_EL1
11	000	0000	0011	000	RO	MVFR0_EL1
11	000	0000	0011	001	RO	MVFR1_EL1
11	000	0000	0011	010	RO	MVFR2_EL1
11	000	0000	0011	100	RO	ID_PFR2_EL1
11	000	0000	0011	101	RO	ID_DFR1_EL1
11	000	0000	0011	110	RO	ID_MMFR5_EL1
11	000	0000	0100	000	RO	ID_AA64PFR0_EL1
11	000	0000	0100	001	RO	ID_AA64PFR1_EL1
11	000	0000	0100	010	RO	ID_AA64PFR2_EL1
11	000	0000	0100	100	RO	ID_AA64ZFR0_EL1
11	000	0000	0100	101	RO	ID_AA64SMFR0_EL1
11	000	0000	0101	000	RO	ID_AA64DFR0_EL1
11	000	0000	0101	001	RO	ID_AA64DFR1_EL1
11	000	0000	0101	100	RO	ID_AA64AFR0_EL1
11	000	0000	0101	101	RO	ID_AA64AFR1_EL1
11	000	0000	0110	000	RO	ID_AA64ISAR0_EL1
11	000	0000	0110	001	RO	ID_AA64ISAR1_EL1
11	000	0000	0110	010	RO	ID_AA64ISAR2_EL1
11	000	0000	0111	000	RO	ID_AA64MMFR0_EL1
11	000	0000	0111	001	RO	ID_AA64MMFR1_EL1
11	000	0000	0111	010	RO	ID_AA64MMFR2_EL1
11	000	0000	0111	011	RO	ID_AA64MMFR3_EL1
11	000	0000	0111	100	RO	ID_AA64MMFR4_EL1
11	000	0001	0000	000	RW	SCTLR_EL1
11	000	0001	0000	000	RW	SCTLR_EL1
11	000	0001	0000	000	RW	SCTLR_EL1
11	000	0001	0000	000	RW	SCTLR_EL1
11	000	0001	0000	001	RW	ACTLR_EL1
11	000	0001	0000	010	RW	CPACR_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	0001	0000	010	RW	CPACR_EL1
11	000	0001	0000	010	RW	CPACR_EL1
11	000	0001	0000	010	RW	CPACR_EL1
11	000	0001	0000	011	RW	SCTLR2_EL1
11	000	0001	0000	011	RW	SCTLR2_EL1
11	000	0001	0000	011	RW	SCTLR2_EL1
11	000	0001	0000	011	RW	SCTLR2_EL1
11	000	0001	0000	101	RW	RGSR_EL1
11	000	0001	0000	110	RW	GCR_EL1
11	000	0001	0010	000	RW	ZCR_EL1
11	000	0001	0010	000	RW	ZCR_EL1
11	000	0001	0010	000	RW	ZCR_EL1
11	000	0001	0010	000	RW	ZCR_EL1
11	000	0001	0010	001	RW	TRFCR_EL1
11	000	0001	0010	001	RW	TRFCR_EL1
11	000	0001	0010	001	RW	TRFCR_EL1
11	000	0001	0010	001	RW	TRFCR_EL1
11	000	0001	0010	010	RW	CCTLR_EL1
11	000	0001	0010	010	RW	CCTLR_EL1
11	000	0001	0010	010	RW	CCTLR_EL1
11	000	0001	0010	010	RW	CCTLR_EL1
11	000	0001	0010	011	RW	TRCITECR_EL1
11	000	0001	0010	011	RW	TRCITECR_EL1
11	000	0001	0010	011	RW	TRCITECR_EL1
11	000	0001	0010	011	RW	TRCITECR_EL1
11	000	0001	0010	100	RW	SMPRI_EL1
11	000	0001	0010	110	RW	SMCR_EL1
11	000	0001	0010	110	RW	SMCR_EL1
11	000	0001	0010	110	RW	SMCR_EL1
11	000	0001	0010	110	RW	SMCR_EL1
11	000	0010	0000	000	RW	TTBR0_EL1
11	000	0010	0000	000	RW	TTBR0_EL1
11	000	0010	0000	000	RW	TTBR0_EL1
11	000	0010	0000	001	RW	TTBR1_EL1
11	000	0010	0000	001	RW	TTBR1_EL1
11	000	0010	0000	001	RW	TTBR1_EL1
11	000	0010	0000	001	RW	TTBR1_EL1
11	000	0010	0000	010	RW	TCR_EL1
11	000	0010	0000	010	RW	TCR_EL1
11	000	0010	0000	010	RW	TCR_EL1
11	000	0010	0000	010	RW	TCR_EL1
11	000	0010	0000	011	RW	TCR2_EL1
11	000	0010	0000	011	RW	TCR2_EL1
11	000	0010	0000	011	RW	TCR2_EL1
11	000	0010	0000	011	RW	TCR2_EL1
11	000	0010	0001	000	RW	APIAKeyLo_EL1
11	000	0010	0001	001	RW	APIAKeyHi_EL1
11	000	0010	0001	010	RW	APIBKeyLo_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	0010	0001	011	RW	APIBKeyHi_EL1
11	000	0010	0010	000	RW	APDAKeyLo_EL1
11	000	0010	0010	001	RW	APDAKeyHi_EL1
11	000	0010	0010	010	RW	APDBKeyLo_EL1
11	000	0010	0010	011	RW	APDBKeyHi_EL1
11	000	0010	0011	000	RW	APGAKeyLo_EL1
11	000	0010	0011	001	RW	APGAKeyHi_EL1
11	000	0010	0101	000	RW	GCSCR_EL1
11	000	0010	0101	000	RW	GCSCR_EL1
11	000	0010	0101	000	RW	GCSCR_EL1
11	000	0010	0101	000	RW	GCSCR_EL1
11	000	0010	0101	001	RW	GCSPR_EL1
11	000	0010	0101	001	RW	GCSPR_EL1
11	000	0010	0101	001	RW	GCSPR_EL1
11	000	0010	0101	001	RW	GCSPR_EL1
11	000	0010	0101	010	RW	GCSCRE0_EL1
11	000	0100	0000	000	RW	SPSR_EL1
11	000	0100	0000	000	RW	SPSR_EL1
11	000	0100	0000	000	RW	SPSR_EL1
11	000	0100	0000	000	RW	SPSR_EL1
11	000	0100	0000	001	RW	ELR_EL1
11	000	0100	0000	001	RW	ELR_EL1
11	000	0100	0000	001	RW	ELR_EL1
11	000	0100	0000	001	RW	ELR_EL1
11	000	0100	0001	000	RW	SP_EL0
11	000	0100	0010	000	-	SPSel
11	000	0100	0010	010	-	CurrentEL
11	000	0100	0010	011	-	PAN
11	000	0100	0010	100	-	UAO
11	000	0100	0011	000	-	ALLINT
11	000	0100	0011	001	-	PM
11	000	0100	0110	000	RW	ICC_PMR_EL1
11	000	0100	0110	000	RW	ICC_PMR_EL1
11	000	0100	0110	000	RW	ICC_PMR_EL1
11	000	0100	0110	000	RW	ICC_PMR_EL1
11	000	0101	0001	000	RW	AFSR0_EL1
11	000	0101	0001	000	RW	AFSR0_EL1
11	000	0101	0001	000	RW	AFSR0_EL1
11	000	0101	0001	000	RW	AFSR0_EL1
11	000	0101	0001	001	RW	AFSR1_EL1
11	000	0101	0001	001	RW	AFSR1_EL1
11	000	0101	0001	001	RW	AFSR1_EL1
11	000	0101	0001	001	RW	AFSR1_EL1
11	000	0101	0010	000	RW	ESR_EL1
11	000	0101	0010	000	RW	ESR_EL1
11	000	0101	0010	000	RW	ESR_EL1
11	000	0101	0010	000	RW	ESR_EL1
11	000	0101	0011	000	RO	ERRIDR_EL1
11	000	0101	0011	001	RW	ERRSELR_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	0101	0011	010	RO	ERXGSR_EL1
11	000	0101	0011	011	RO	DCRPZID_EL1
11	000	0101	0100	000	RO	ERXFR_EL1
11	000	0101	0100	001	RW	ERXCTLR_EL1
11	000	0101	0100	010	RW	ERXSTATUS_EL1
11	000	0101	0100	011	RW	ERXADDR_EL1
11	000	0101	0100	100	RO	ERXPFGF_EL1
11	000	0101	0100	101	RW	ERXPFGCTL_EL1
11	000	0101	0100	110	RW	ERXPFGCDN_EL1
11	000	0101	0101	000	RW	ERXMISC0_EL1
11	000	0101	0101	001	RW	ERXMISC1_EL1
11	000	0101	0101	010	RW	ERXMISC2_EL1
11	000	0101	0101	011	RW	ERXMISC3_EL1
11	000	0101	0110	000	RW	TFSR_EL1
11	000	0101	0110	000	RW	TFSR_EL1
11	000	0101	0110	000	RW	TFSR_EL1
11	000	0101	0110	000	RW	TFSR_EL1
11	000	0101	0110	001	RW	TFSRE0_EL1
11	000	0110	0000	000	RW	FAR_EL1
11	000	0110	0000	000	RW	FAR_EL1
11	000	0110	0000	000	RW	FAR_EL1
11	000	0110	0000	000	RW	FAR_EL1
11	000	0110	0001	001	RW	PRENR_EL1
11	000	0110	0010	001	RW	PRSELR_EL1
11	000	0110	1000	000	RW	PRBAR_EL1
11	000	0110	1000	001	RW	PRLAR_EL1
11	000	0110	1:m[3:1]	m[0]:00	RW	PRBAR<m>_EL1
11	000	0110	1:m[3:1]	m[0]:01	RW	PRLAR<m>_EL1
11	000	0110	0000	101110	RW	PFAR_EL1
11	000	0110	0000	101110	RW	PFAR_EL1
11	000	0111	0100	000	RW	PAR_EL1
11	000	1001	1001	000	RW	PMSCR_EL1
11	000	1001	1001	000	RW	PMSCR_EL1
11	000	1001	1001	000	RW	PMSCR_EL1
11	000	1001	1001	000	RW	PMSCR_EL1
11	000	1001	1001	001	RW	PMSNEVER_EL1
11	000	1001	1001	010	RW	PMSICR_EL1
11	000	1001	1001	011	RW	PMSIRR_EL1
11	000	1001	1001	100	RW	PMSECR_EL1
11	000	1001	1001	101	RW	PMSEVER_EL1
11	000	1001	1001	110	RW	PMSLATER_EL1
11	000	1001	1001	111	RO	PMSIDR_EL1
11	000	1001	1010	000	RW	PMBLIMITR_EL1
11	000	1001	1010	001	RW	PMBPTR_EL1
11	000	1001	1010	011	RW	PMBSR_EL1
11	000	1001	1010	100	RW	PMSDSFR_EL1
11	000	1001	1010	111	RO	PMBIDR_EL1
11	000	1001	1011	000	RW	TRBLIMITR_EL1
11	000	1001	1011	001	RW	TRBPTR_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	1001	1011	010	RW	TRBBASER_EL1
11	000	1001	1011	011	RW	TRBSR_EL1
11	000	1001	1011	100	RW	TRBMAR_EL1
11	000	1001	1011	110	RW	TRBTRG_EL1
11	000	1001	1011	111	RO	TRBIDR_EL1
11	000	1001	1101	011	RW	PMSSCR_EL1
11	000	1001	1110	001	RW	PMINTENSET_EL1
11	000	1001	1110	010	RW	PMINTENCLR_EL1
11	000	1001	1110	100	RW	PMUACR_EL1
11	000	1001	1110	101	RW	PMECR_EL1
11	000	1001	1110	110	RO	PMMIR_EL1
11	000	1001	1110	111	RW	PMIAR_EL1
11	000	1010	0010	000	RW	MAIR_EL1
11	000	1010	0010	000	RW	MAIR_EL1
11	000	1010	0010	000	RW	MAIR_EL1
11	000	1010	0010	000	RW	MAIR_EL1
11	000	1010	0010	001	RW	MAIR2_EL1
11	000	1010	0010	001	RW	MAIR2_EL1
11	000	1010	0010	001	RW	MAIR2_EL1
11	000	1010	0010	001	RW	MAIR2_EL1
11	000	1010	0010	001	RW	MAIR2_EL1
11	000	1010	0010	010	RW	PIRE0_EL1
11	000	1010	0010	010	RW	PIRE0_EL1
11	000	1010	0010	010011	RW	PIRE0_EL1 PIR_EL1
11	000	1010	0010	010	RW	PIRE0_EL1
11	000	1010	0010	010	RW	PIRE0_EL1
11	000	1010	0010	011	RW	PIR_EL1
11	000	1010	0010	011	RW	PIR_EL1
11	000	1010	0010	011	RW	PIR_EL1
11	000	1010	0010	100	RW	POR_EL1
11	000	1010	0010	100	RW	POR_EL1
11	000	1010	0010	100	RW	POR_EL1
11	000	1010	0010	100	RW	POR_EL1
11	000	1010	0010	101	RW	S2POR_EL1
11	000	1010	0011	000	RW	AMAIR_EL1
11	000	1010	0011	000	RW	AMAIR_EL1
11	000	1010	0011	000	RW	AMAIR_EL1
11	000	1010	0011	000	RW	AMAIR_EL1
11	000	1010	0011	001	RW	AMAIR2_EL1
11	000	1010	0011	001	RW	AMAIR2_EL1
11	000	1010	0011	001	RW	AMAIR2_EL1
11	000	1010	0011	001	RW	AMAIR2_EL1
11	000	1010	0100	000	RW	LORSA_EL1
11	000	1010	0100	001	RW	LOREA_EL1
11	000	1010	0100	010	RW	LORN_EL1
11	000	1010	0100	011	RW	LORC_EL1
11	000	1010	0100	100	RO	MPAMIDR_EL1
11	000	1010	0100	111	RO	LORID_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	1010	0101	000	RW	MPAM1_EL1
11	000	1010	0101	000	RW	MPAM1_EL1
11	000	1010	0101	000	RW	MPAM1_EL1
11	000	1010	0101	000	RW	MPAM1_EL1
11	000	1010	0101	001	RW	MPAM0_EL1
11	000	1010	0101	011	RW	MPAMSM_EL1
11	000	1100	0000	000	RW	VBAR_EL1
11	000	1100	0000	000	RW	VBAR_EL1
11	000	1100	0000	000	RW	VBAR_EL1
11	000	1100	0000	000	RW	VBAR_EL1
11	000	1100	0000	001	RO	RVBAR_EL1
11	000	1100	0000	010	RW	RMR_EL1
11	000	1100	0001	000	RO	ISR_EL1
11	000	1100	0001	001	RW	DISR_EL1
11	000	1100	0001	001	RW	DISR_EL1
11	000	1100	0001	001	RW	DISR_EL1
11	000	1100	0001	001	RW	DISR_EL1
11	000	1100	1000	000	RO	ICC_IAR0_EL1
11	000	1100	1000	000	RO	ICC_IAR0_EL1
11	000	1100	1000	000	RO	ICC_IAR0_EL1
11	000	1100	1000	000	RO	ICC_IAR0_EL1
11	000	1100	1000	001	WO	ICC_EOIR0_EL1
11	000	1100	1000	001	WO	ICC_EOIR0_EL1
11	000	1100	1000	001	WO	ICC_EOIR0_EL1
11	000	1100	1000	001	WO	ICC_EOIR0_EL1
11	000	1100	1000	010	RO	ICC_HPPIR0_EL1
11	000	1100	1000	010	RO	ICC_HPPIR0_EL1
11	000	1100	1000	010	RO	ICC_HPPIR0_EL1
11	000	1100	1000	010	RO	ICC_HPPIR0_EL1
11	000	1100	1000	011	RW	ICC_BPR0_EL1
11	000	1100	1000	011	RW	ICC_BPR0_EL1
11	000	1100	1000	011	RW	ICC_BPR0_EL1
11	000	1100	1000	011	RW	ICC_BPR0_EL1
11	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>_EL1
11	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>_EL1
11	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>_EL1
11	000	1100	1000	1:m[1:0]	RW	ICC_AP0R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	0:m[1:0]	RW	ICC_AP1R<m>_EL1
11	000	1100	1001	101	RO	ICC_NMIAR1_EL1
11	000	1100	1001	101	RO	ICC_NMIAR1_EL1
11	000	1100	1001	101	RO	ICC_NMIAR1_EL1
11	000	1100	1001	101	RO	ICC_NMIAR1_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	1100	1011	001	WO	ICC_DIR_EL1
11	000	1100	1011	001	WO	ICC_DIR_EL1
11	000	1100	1011	001	WO	ICC_DIR_EL1
11	000	1100	1011	001	WO	ICC_DIR_EL1
11	000	1100	1011	011	RO	ICC_RPR_EL1
11	000	1100	1011	011	RO	ICC_RPR_EL1
11	000	1100	1011	011	RO	ICC_RPR_EL1
11	000	1100	1011	011	RO	ICC_RPR_EL1
11	000	1100	1011	101	WO	ICC_SGI1R_EL1
11	000	1100	1011	110	WO	ICC_ASGI1R_EL1
11	000	1100	1011	111	WO	ICC_SGI0R_EL1
11	000	1100	1100	000	RO	ICC_IAR1_EL1
11	000	1100	1100	000	RO	ICC_IAR1_EL1
11	000	1100	1100	000	RO	ICC_IAR1_EL1
11	000	1100	1100	000	RO	ICC_IAR1_EL1
11	000	1100	1100	001	WO	ICC_EOIR1_EL1
11	000	1100	1100	001	WO	ICC_EOIR1_EL1
11	000	1100	1100	001	WO	ICC_EOIR1_EL1
11	000	1100	1100	001	WO	ICC_EOIR1_EL1
11	000	1100	1100	010	RO	ICC_HPPIR1_EL1
11	000	1100	1100	010	RO	ICC_HPPIR1_EL1
11	000	1100	1100	010	RO	ICC_HPPIR1_EL1
11	000	1100	1100	010	RO	ICC_HPPIR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	011	RW	ICC_BPR1_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	100	RW	ICC_CTLR_EL1
11	000	1100	1100	101	RW	ICC_SRE_EL1
11	000	1100	1100	101	RW	ICC_SRE_EL1
11	000	1100	1100	101	RW	ICC_SRE_EL1
11	000	1100	1100	110	RW	ICC_IGRPEN0_EL1
11	000	1100	1100	110	RW	ICC_IGRPEN0_EL1
11	000	1100	1100	110	RW	ICC_IGRPEN0_EL1
11	000	1100	1100	110	RW	ICC_IGRPEN0_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1100	1100	111	RW	ICC_IGRPEN1_EL1
11	000	1101	0000	001	RW	CONTEXTIDR_EL1
11	000	1101	0000	001	RW	CONTEXTIDR_EL1
11	000	1101	0000	001	RW	CONTEXTIDR_EL1
11	000	1101	0000	001	RW	CONTEXTIDR_EL1
11	000	1101	0000	011	RW	RCWSMASK_EL1 SOFTRCWM
11	000	1101	0000	100	RW	TPIDR_EL1
11	000	1101	0000	100	RW	TPIDR_EL1
11	000	1101	0000	101	RW	ACCDATA_EL1
11	000	1101	0000	110	RW	RCWMASK_EL1
11	000	1101	0000	111	RW	SCXTNUM_EL1
11	000	1101	0000	111	RW	SCXTNUM_EL1
11	000	1101	0000	111	RW	SCXTNUM_EL1
11	000	1101	0000	111	RW	SCXTNUM_EL1
11	000	1110	0001	000	RW	CNTKCTL_EL1
11	000	1110	0001	000	RW	CNTKCTL_EL1
11	000	1110	0001	000	RW	CNTKCTL_EL1
11	000	1110	0001	000	RW	CNTKCTL_EL1
11	001	0000	0000	000	RO	CCSIDR_EL1
11	001	0000	0000	001	RO	CLIDR_EL1
11	001	0000	0000	010	RO	CCSIDR2_EL1
11	001	0000	0000	100	RO	GMID_EL1
11	001	0000	0000	110	RO	SMIDR_EL1
11	001	0000	0000	111	RO	AIDR_EL1
11	010	0000	0000	000	RW	CSSELR_EL1
11	011	0000	0000	001	RO	CTR_EL0
11	011	0000	0000	111	RO	DCZID_EL0
11	011	0001	0010	010	RW	CCTLR_EL0
11	011	0010	0100	000	RO	RNDR
11	011	0010	0100	001	RO	RNDRRS
11	011	0010	0101	001	RW	GCSPR_EL0
11	011	0100	0010	000	-	NZCV
11	011	0100	0010	001	-	DAIF
11	011	0100	0010	010	-	SVCR
11	011	0100	0010	101	-	DIT
11	011	0100	0010	110	-	SSBS
11	011	0100	0010	111	-	TCO
11	011	0100	0100	000	RW	FPCR
11	011	0100	0100	001	RW	FPSR
11	011	0100	0101	000	RW	DSPSR_EL0
11	011	0100	0101	001	RW	DLR_EL0
11	011	1001	0100	000	RW	PMICNTR_EL0
11	011	1001	0110	000	RW	PMICFILTR_EL0
11	011	1001	1100	000	RW	PMCRR_EL0
11	011	1001	1100	001	RW	PMCNTENSET_EL0

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	011	1001	1100	010	RW	PMCNTENCLR_ELO
11	011	1001	1100	011	RW	PMOVSCLR_ELO
11	011	1001	1100	100	WO	PMSWINC_ELO
11	011	1001	1100	101	RW	PMSELR_ELO
11	011	1001	1100	110	RO	PMCEID0_ELO
11	011	1001	1100	111	RO	PMCEID1_ELO
11	011	1001	1101	000	RW	PMCCNTR_ELO
11	011	1001	1101	001	RW	PMXEVTYPER_ELO
11	011	1001	1101	001	RW	PMXEVTYPER_ELO
11	011	1001	1101	010	RW	PMXEVCNTR_ELO
11	011	1001	1101	100	WO	PMZR_ELO
11	011	1001	1110	000	RW	PMUSERENR_ELO
11	011	1001	1110	011	RW	PMOVSSET_ELO
11	011	1010	0010	100	RW	POR_ELO
11	011	1101	0000	010	RW	TPIDR_ELO
11	011	1101	0000	010	RW	TPIDR_ELO
11	011	1101	0000	011	RW	TPIDRRO_ELO
11	011	1101	0000	100	RW	RTPIDR_ELO
11	011	1101	0000	101	RW	TPIDR2_ELO
11	011	1101	0000	111	RW	SCXTNUM_ELO
11	011	1101	0010	000	RW	AMCR_ELO
11	011	1101	0010	001	RO	AMCFGR_ELO
11	011	1101	0010	010	RO	AMCGCR_ELO
11	011	1101	0010	011	RW	AMUSERENR_ELO
11	011	1101	0010	100	RW	AMCNTENCLR0_ELO
11	011	1101	0010	101	RW	AMCNTENSET0_ELO
11	011	1101	0010	110	RO	AMCG1IDR_ELO
11	011	1101	0011	000	RW	AMCNTENCLR1_ELO
11	011	1101	0011	001	RW	AMCNTENSET1_ELO
11	011	1101	010:m[3]	m[2:0]	RW	AMEVCNTR0<m>_ELO
11	011	1101	011:m[3]	m[2:0]	RO	AMEVTYPER0<m>_ELO
11	011	1101	110:m[3]	m[2:0]	RW	AMEVCNTR1<m>_ELO
11	011	1101	111:m[3]	m[2:0]	RW	AMEVTYPER1<m>_ELO
11	011	1110	0000	000	RW	CNTFRQ_ELO
11	011	1110	0000	001	-	CNTPCT_ELO
11	011	1110	0000	010	-	CNTVCT_ELO
11	011	1110	0000	101	-	CNTPCTSS_ELO
11	011	1110	0000	110	-	CNTVCTSS_ELO
11	011	1110	0010	000	-	CNTP_TVAL_ELO
11	011	1110	0010	000	-	CNTP_TVAL_ELO
11	011	1110	0010	000	-	CNTP_TVAL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO
11	011	1110	0010	001	RW	CNTP_CTL_ELO

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	011	1110	0010	001	RW	CNTP_CTL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0010	010	RW	CNTP_CVAL_EL0
11	011	1110	0011	000	-	CNTV_TVAL_EL0
11	011	1110	0011	000	-	CNTV_TVAL_EL0
11	011	1110	0011	000	-	CNTV_TVAL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	001	RW	CNTV_CTL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	0011	010	RW	CNTV_CVAL_EL0
11	011	1110	10:m[4:3]	m[2:0]	RW	PMEVCNTR<m>_EL0
11	011	1110	1111	111	RW	PMCCFILTR_EL0
11	011	1110	11:m[4:3]	m[2:0]	RW	PMEVTPER<m>_EL0
11	100	0000	0000	000	RO	VPIDR_EL2
11	100	0000	0000	000	RW	VPIDR_EL2
11	100	0000	0000	100	RO	MPUIR_EL2
11	100	0000	0000	101	RO	VMPIDR_EL2
11	100	0000	0000	101	RW	VMPIDR_EL2
11	100	0001	0000	000	RW	SCTLR_EL2
11	100	0001	0000	001	RW	ACTLR_EL2
11	100	0001	0000	011	RW	SCTLR2_EL2
11	100	0001	0001	000	RW	HCR_EL2
11	100	0001	0001	001	RW	MDCR_EL2
11	100	0001	0001	010	RW	CPTR_EL2
11	100	0001	0001	011	RW	HSTR_EL2
11	100	0001	0001	100	RW	HFGTR_EL2
11	100	0001	0001	101	RW	HFGWTR_EL2
11	100	0001	0001	110	RW	HFGITR_EL2

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	100	0001	0001	111	RW	HACR_EL2
11	100	0001	0010	000	RW	ZCR_EL2
11	100	0001	0010	001	RW	TRFCR_EL2
11	100	0001	0010	010	RW	CCTLR_EL2
11	100	0001	0010	010	RW	HCRX_EL2
11	100	0001	0010	011	RW	CHCR_EL2
11	100	0001	0010	011	RW	TRCITECR_EL2
11	100	0001	0010	101	RW	SMPRIMAP_EL2
11	100	0001	0010	110	RW	SMCR_EL2
11	100	0001	0011	001	RW	SDER32_EL2
11	100	0010	0000	000	RW	TTBR0_EL2
11	100	0010	0000	000	RW	VSCTLR_EL2
11	100	0010	0000	001	RW	TTBR1_EL2
11	100	0010	0000	010	RW	TCR_EL2
11	100	0010	0000	011	RW	TCR2_EL2
11	100	0010	0001	000	RW	VTTBR_EL2
11	100	0010	0001	010	RW	VTCR_EL2
11	100	0010	0010	000	RW	VNCR_EL2
11	100	0010	0101	000	RW	GCSCR_EL2
11	100	0010	0101	001	RW	GCSPR_EL2
11	100	0010	0110	000	RW	VSTTBR_EL2
11	100	0010	0110	010	RW	VSTCR_EL2
11	100	0011	0000	000	RW	DACR32_EL2
11	100	0011	0001	000	RW	HDFGRTR2_EL2
11	100	0011	0001	001	RW	HDFGWTR2_EL2
11	100	0011	0001	010	RW	HFGRTR2_EL2
11	100	0011	0001	011	RW	HFGWTR2_EL2
11	100	0011	0001	100	RW	HDFGRTR_EL2
11	100	0011	0001	101	RW	HDFGWTR_EL2
11	100	0011	0001	110	RW	HAFGRTR_EL2
11	100	0011	0001	111	RW	HFGITR2_EL2
11	100	0100	0000	000	RW	SPSR_EL2
11	100	0100	0000	000	RW	SPSR_EL2
11	100	0100	0000	000	RW	SPSR_EL2
11	100	0100	0000	000	RW	SPSR_EL2
11	100	0100	0000	001	RW	ELR_EL2
11	100	0100	0000	001	RW	ELR_EL2
11	100	0100	0000	001	RW	ELR_EL2
11	100	0100	0000	001	RW	ELR_EL2
11	100	0100	0001	000	RW	SP_EL1
11	100	0100	0011	000	RW	SPSR_irq
11	100	0100	0011	001	RW	SPSR_abt
11	100	0100	0011	010	RW	SPSR_und
11	100	0100	0011	011	RW	SPSR_fiq
11	100	0101	0000	001	RW	IFSR32_EL2
11	100	0101	0001	000	RW	AFSR0_EL2
11	100	0101	0001	001	RW	AFSR1_EL2
11	100	0101	0010	000	RW	ESR_EL2
11	100	0101	0010	000	RW	ESR_EL2

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	100	0101	0010	000	RW	ESR_EL2
11	100	0101	0010	000	RW	ESR_EL2
11	100	0101	0010	011	RW	VSESR_EL2
11	100	0101	0011	000	RW	FPEXC32_EL2
11	100	0101	0110	000	RW	TFSR_EL2
11	100	0101	0110	000	RW	TFSR_EL2
11	100	0101	0110	000	RW	TFSR_EL2
11	100	0101	0110	000	RW	TFSR_EL2
11	100	0110	0000	000	RW	FAR_EL2
11	100	0110	0000	000	RW	FAR_EL2
11	100	0110	0000	000	RW	FAR_EL2
11	100	0110	0000	000	RW	FAR_EL2
11	100	0110	0000	100	RW	HPFAR_EL2
11	100	0110	0001	001	RW	PRENR_EL2
11	100	0110	0010	001	RW	PRSELR_EL2
11	100	0110	1000	000	RW	PRBAR_EL2
11	100	0110	1000	001	RW	PRLAR_EL2
11	100	0110	1:m[3:1]	m[0]:00	RW	PRBAR<m>_EL2
11	100	0110	1:m[3:1]	m[0]:01	RW	PRLAR<m>_EL2
11	100	0110	0000	101110	RW	PFAR_EL2
11	100	1001	1001	000	RW	PMSCR_EL2
11	100	1010	0001	001	RW	MAIR2_EL2
11	100	1010	0010	000	RW	MAIR_EL2
11	100	1010	0010	010	RW	PIRE0_EL2
11	100	1010	0010	011	RW	PIR_EL2
11	100	1010	0010	100	RW	POR_EL2
11	100	1010	0010	101	RW	S2PIR_EL2
11	100	1010	0011	000	RW	AMAIR_EL2
11	100	1010	0011	001	RW	AMAIR2_EL2
11	100	1010	0100	000	RW	MPAMHCR_EL2
11	100	1010	0100	001	RW	MPAMVPMV_EL2
11	100	1010	0101	000	RW	MPAM2_EL2
11	100	1010	0110	000	RW	MPAMVPM0_EL2
11	100	1010	0110	001	RW	MPAMVPM1_EL2
11	100	1010	0110	010	RW	MPAMVPM2_EL2
11	100	1010	0110	011	RW	MPAMVPM3_EL2
11	100	1010	0110	100	RW	MPAMVPM4_EL2
11	100	1010	0110	101	RW	MPAMVPM5_EL2
11	100	1010	0110	110	RW	MPAMVPM6_EL2
11	100	1010	0110	111	RW	MPAMVPM7_EL2
11	100	10101011	10000000	000	RO	MECID_P0_EL2 MECID_P_EL2
11	100	10101011	10000000	001	RO	MECID_A0_EL2 MECID_A_EL2
11	100	10101011	10000000	010111	RO	MECID_P1_EL2 MECIDR_EL2
11	100	10101011	10000001	011000	RO	MECID_A1_EL2 VMECID_P_EL2
11	100	10101011	10000001	111001	RO	MECIDR_EL2 VMECID_A_EL2
11	100	1010	1001	000	RO	VMECID_P_EL2
11	100	1010	1001	001	RO	VMECID_A_EL2
11	100	1100	0000	000	RW	VBAR_EL2
11	100	1100	0000	001	RO	RVBAR_EL2

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	100	1100	0000	010	RW	RMR_EL2
11	100	1100	0001	001	RW	VDISR_EL2
11	100	1100	1000	0:m[1:0]	RW	ICH_AP0R<m>_EL2
11	100	1100	1001	0:m[1:0]	RW	ICH_AP1R<m>_EL2
11	100	1100	1001	101	RW	ICC_SRE_EL2
11	100	1100	1011	000	RW	ICH_HCR_EL2
11	100	1100	1011	001	RO	ICH_VTR_EL2
11	100	1100	1011	010	RO	ICH_MISR_EL2
11	100	1100	1011	011	RO	ICH_EISR_EL2
11	100	1100	1011	101	RO	ICH_ELRSR_EL2
11	100	1100	1011	111	RW	ICH_VMCR_EL2
11	100	1100	110:m[3]	m[2:0]	RW	ICH_LR<m>_EL2
11	100	1101	0000	001	RW	CONTEXTIDR_EL2
11	100	1101	0000	010	RW	TPIDR_EL2
11	100	1101	0000	010	RW	TPIDR_EL2
11	100	1101	0000	111	RW	SCXTNUM_EL2
11	100	1101	100:m[3]	m[2:0]	RW	AMEVCNTVOFF0<m>_EL2
11	100	1101	101:m[3]	m[2:0]	RW	AMEVCNTVOFF1<m>_EL2
11	100	1110	0000	011	RW	CNTVOFF_EL2
11	100	1110	0000	110	RW	CNTPOFF_EL2
11	100	1110	0001	000	RW	CNTHCTL_EL2
11	100	1110	0010	000	-	CNTHP_TVAL_EL2
11	100	1110	0010	001	RW	CNTHP_CTL_EL2
11	100	1110	0010	010	RW	CNTHP_CVAL_EL2
11	100	1110	0011	000	-	CNTHV_TVAL_EL2
11	100	1110	0011	001	RW	CNTHV_CTL_EL2
11	100	1110	0011	010	RW	CNTHV_CVAL_EL2
11	100	1110	0100	000	-	CNTHVS_TVAL_EL2
11	100	1110	0100	001	RW	CNTHVS_CTL_EL2
11	100	1110	0100	010	RW	CNTHVS_CVAL_EL2
11	100	1110	0101	000	-	CNTHPS_TVAL_EL2
11	100	1110	0101	001	RW	CNTHPS_CTL_EL2
11	100	1110	0101	010	RW	CNTHPS_CVAL_EL2
11	101	0001	0000	000	RW	SCTLR_EL12
11	101	0001	0000	010	RW	CPACR_EL12
11	101	0001	0000	011	RW	SCTLR2_EL12
11	101	0001	0010	000	RW	ZCR_EL12
11	101	0001	0010	001	RW	TRFCR_EL12
11	101	0001	0010	010	RW	CCTLR_EL12
11	101	0001	0010	011	RW	TRCITECR_EL12
11	101	0001	0010	110	RW	SMCR_EL12
11	101	0010	0000	000	RW	TTBR0_EL12
11	101	0010	0000	001	RW	TTBR1_EL12
11	101	0010	0000	010	RW	TCR_EL12
11	101	0010	0101	000	RW	GCSCR_EL12
11	101	0010	0101	001	RW	GCSPR_EL12
11	101	0010	0000	011	RW	TCR2_EL12
11	101	0100	0000	000	RW	SPSR_EL12
11	101	0100	0000	001	RW	ELR_EL12

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	101	0101	0001	000	RW	AFSR0_EL12
11	101	0101	0001	001	RW	AFSR1_EL12
11	101	0101	0010	000	RW	ESR_EL12
11	101	0101	0110	000	RW	TFSR_EL12
11	101	0110	0000	000	RW	FAR_EL12
11	101	0110	0000	101110	RW	PFAR_EL12
11	101	1001	1001	000	RW	PMSCR_EL12
11	101	1010	0010	000	RW	MAIR_EL12
11	101	1010	0010	001	RW	MAIR2_EL12
11	101	1010	0010	010	RW	PIRE0_EL12
11	101	1010	0010	011	RW	PIR_EL12
11	101	1010	0010	100	RW	POR_EL12
11	101	1010	0011	000	RW	AMAIR_EL12
11	101	1010	0011	001	RW	AMAIR2_EL12
11	101	1010	0101	000	RW	MPAM1_EL12
11	101	1100	0000	000	RW	VBAR_EL12
11	101	1101	0000	001	RW	CONTEXTIDR_EL12
11	101	1101	0000	111	RW	SCXTNUM_EL12
11	101	1110	0001	000	RW	CNTKCTL_EL12
11	101	1110	0010	000	-	CNTP_TVAL_EL02
11	101	1110	0010	001	RW	CNTP_CTL_EL02
11	101	1110	0010	010	RW	CNTP_CVAL_EL02
11	101	1110	0011	000	-	CNTV_TVAL_EL02
11	101	1110	0011	001	RW	CNTV_CTL_EL02
11	101	1110	0011	010	RW	CNTV_CVAL_EL02
11	110	0001	0000	000	RW	SCTLR_EL3
11	110	0001	0000	001	RW	ACTLR_EL3
11	110	0001	0000	011	RW	SCTLR2_EL3
11	110	0001	0001	000	RW	SCR_EL3
11	110	0001	0001	001	RW	SDER32_EL3
11	110	0001	0001	010	RW	CPTR_EL3
11	110	0001	0010	000	RW	ZCR_EL3
11	110	0001	0010	010	RW	CCTLR_EL3
11	110	0001	0010	011	RW	CSCR_EL3
11	110	0001	0010	110	RW	SMCR_EL3
11	110	0001	0011	001	RW	MDCR_EL3
11	110	0010	0000	000	RW	TTBR0_EL3
11	110	0010	0000	010	RW	TCR_EL3
11	110	0010	0001	100	RW	GPTBR_EL3
11	110	0010	0001	110	RW	GPCCR_EL3
11	110	0010	0101	000	RW	GCSCR_EL3
11	110	0010	0101	001	RW	GCSPR_EL3
11	110	0100	0000	000	RW	SPSR_EL3
11	110	0100	0000	001	RW	ELR_EL3
11	110	0100	0001	000	RW	SP_EL2
11	110	0101	0001	000	RW	AFSR0_EL3
11	110	0101	0001	001	RW	AFSR1_EL3
11	110	0101	0010	000	RW	ESR_EL3
11	110	0101	0110	000	RW	TFSR_EL3

op0	op1	CRn	CRm	op2	Access	Mnemonic
11	110	0110	0000	000	RW	FAR_EL3
11	110	0110	0000	101	RW	MFAR_EL3
11	110	0110	0000	110	RW	PFAR_EL3
11	110	1010	0001	001	RW	MAIR2_EL3
11	110	1010	0010	000	RW	MAIR_EL3
11	110	1010	0010	011	RW	PIR_EL3
11	110	1010	0010	100	RW	POR_EL3
11	110	1010	0011	000	RW	AMAIR_EL3
11	110	1010	0011	001	RW	AMAIR2_EL3
11	110	1010	0101	000	RW	MPAM3_EL3
11	110	10101011	10100010	001	RO	MECID_RL_A_EL3
11	110	1100	0000	000	RW	VBAR_EL3
11	110	1100	0000	001	RO	RVBAR_EL3
11	110	1100	0000	010	RW	RMR_EL3
11	110	1100	1100	100	RW	ICC_CTLR_EL3
11	110	1100	1100	101	RW	ICC_SRE_EL3
11	110	1100	1100	111	RW	ICC_IGRPEN1_EL3
11	110	1101	0000	010	RW	TPIDR_EL3
11	110	1101	0000	010	RW	TPIDR_EL3
11	110	1101	0000	111	RW	SCXTNUM_EL3
11	111	0100	0001	011	RW	RSP_EL0
11	111	1110	0010	000	-	CNTPS_TVAL_EL1
11	111	1110	0010	001	RW	CNTPS_CTL_EL1
11	111	1110	0010	010	RW	CNTPS_CVAL_EL1
11	op1[2:0]	1x11	Cm[3:0]	op2[2:0]	-	S3_<op1>_C<Cn>_C<Cm>_<op2>

Accessed using MRRS/MSRR:

op0	op1	CRn	CRm	op2	Mnemonic
11	000	0010	0000	000	TTBR0_EL1
11	000	0010	0000	000	TTBR0_EL1
11	000	0010	0000	001	TTBR1_EL1
11	000	0010	0000	001	TTBR1_EL1
11	000	0111	0100	000	PAR_EL1
11	000	1101	0000	011	RCWSMASK_EL1
11	000	1101	0000	110	RCWSMASK_EL1
11	100	0010	0000	000	TTBR0_EL2
11	100	0010	0000	001	TTBR1_EL2
11	100	0010	0001	000	VTTBR_EL2
11	101	0010	0000	000	TTBR0_EL12
11	101	0010	0000	001	TTBR1_EL12
11	op1[2:0]	1x11	Cm[3:0]	op2[2:0]	S3_<op1>_C<Cn>_C<Cm>_<op2>

Accessed using TLBI:

op0	op1	CRn	CRm	op2	Mnemonic
01	000	1000	0001	000	TLBI_VMALE1OS
01	000	1000	0001	001	TLBI_VAE1OS
01	000	1000	0001	010	TLBI_ASIDE1OS

op0	op1	CRn	CRm	op2	Mnemonic
01	000	1000	0001	011	TLBI VAAE1OS
01	000	1000	0001	101	TLBI VALE1OS
01	000	1000	0001	111	TLBI VAALE1OS
01	000	1000	0010	001	TLBI RVAE1IS
01	000	1000	0010	011	TLBI RVAAE1IS
01	000	1000	0010	101	TLBI RVALE1IS
01	000	1000	0010	111	TLBI RVAALE1IS
01	000	1000	0011	000	TLBI VMALLE1IS
01	000	1000	0011	001	TLBI VAE1IS
01	000	1000	0011	010	TLBI ASIDE1IS
01	000	1000	0011	011	TLBI VAAE1IS
01	000	1000	0011	101	TLBI VALE1IS
01	000	1000	0011	111	TLBI VAALE1IS
01	000	1000	0101	001	TLBI RVAE1OS
01	000	1000	0101	011	TLBI RVAAE1OS
01	000	1000	0101	101	TLBI RVALE1OS
01	000	1000	0101	111	TLBI RVAALE1OS
01	000	1000	0110	001	TLBI RVAE1
01	000	1000	0110	011	TLBI RVAAE1
01	000	1000	0110	101	TLBI RVALE1
01	000	1000	0110	111	TLBI RVAALE1
01	000	1000	0111	000	TLBI VMALLE1
01	000	1000	0111	001	TLBI VAE1
01	000	1000	0111	010	TLBI ASIDE1
01	000	1000	0111	011	TLBI VAAE1
01	000	1000	0111	101	TLBI VALE1
01	000	1000	0111	111	TLBI VAALE1
01	000	1001	0001	000	TLBI VMALLE1OSNXS
01	000	1001	0001	001	TLBI VAE1OSNXS
01	000	1001	0001	010	TLBI ASIDE1OSNXS
01	000	1001	0001	011	TLBI VAAE1OSNXS
01	000	1001	0001	101	TLBI VALE1OSNXS
01	000	1001	0001	111	TLBI VAALE1OSNXS
01	000	1001	0010	001	TLBI RVAE1ISNXS
01	000	1001	0010	011	TLBI RVAAE1ISNXS
01	000	1001	0010	101	TLBI RVALE1ISNXS
01	000	1001	0010	111	TLBI RVAALE1ISNXS
01	000	1001	0011	000	TLBI VMALLE1ISNXS
01	000	1001	0011	001	TLBI VAE1ISNXS
01	000	1001	0011	010	TLBI ASIDE1ISNXS
01	000	1001	0011	011	TLBI VAAE1ISNXS
01	000	1001	0011	101	TLBI VALE1ISNXS
01	000	1001	0011	111	TLBI VAALE1ISNXS
01	000	1001	0101	001	TLBI RVAE1OSNXS
01	000	1001	0101	011	TLBI RVAAE1OSNXS
01	000	1001	0101	101	TLBI RVALE1OSNXS
01	000	1001	0101	111	TLBI RVAALE1OSNXS
01	000	1001	0110	001	TLBI RVAE1NXS
01	000	1001	0110	011	TLBI RVAAE1NXS

op0	op1	CRn	CRm	op2	Mnemonic
01	000	1001	0110	101	TLBI RVALE1NXS
01	000	1001	0110	111	TLBI RVAALE1NXS
01	000	1001	0111	000	TLBI VMALLE1NXS
01	000	1001	0111	001	TLBI VAE1NXS
01	000	1001	0111	010	TLBI ASIDE1NXS
01	000	1001	0111	011	TLBI VAAE1NXS
01	000	1001	0111	101	TLBI VALE1NXS
01	000	1001	0111	111	TLBI VAALE1NXS
01	100	1000	0000	001	TLBI IPAS2E1IS
01	100	1000	0000	010	TLBI RIPAS2E1IS
01	100	1000	0000	101	TLBI IPAS2LE1IS
01	100	1000	0000	110	TLBI RIPAS2LE1IS
01	100	1000	0001	000	TLBI ALLE2OS
01	100	1000	0001	001	TLBI VAE2OS
01	100	1000	0001	100	TLBI ALLE1OS
01	100	1000	0001	101	TLBI VALE2OS
01	100	1000	0001	110	TLBI VMALLS12E1OS
01	100	1000	0010	001	TLBI RVAE2IS
01	100	1000	0010	101	TLBI RVALE2IS
01	100	1000	0011	000	TLBI ALLE2IS
01	100	1000	0011	001	TLBI VAE2IS
01	100	1000	0011	100	TLBI ALLE1IS
01	100	1000	0011	101	TLBI VALE2IS
01	100	1000	0011	110	TLBI VMALLS12E1IS
01	100	1000	0100	000	TLBI IPAS2E1OS
01	100	1000	0100	001	TLBI IPAS2E1
01	100	1000	0100	010	TLBI RIPAS2E1
01	100	1000	0100	011	TLBI RIPAS2E1OS
01	100	1000	0100	100	TLBI IPAS2LE1OS
01	100	1000	0100	101	TLBI IPAS2LE1
01	100	1000	0100	110	TLBI RIPAS2LE1
01	100	1000	0100	111	TLBI RIPAS2LE1OS
01	100	1000	0101	001	TLBI RVAE2OS
01	100	1000	0101	101	TLBI RVALE2OS
01	100	1000	0110	001	TLBI RVAE2
01	100	1000	0110	101	TLBI RVALE2
01	100	1000	0111	000	TLBI ALLE2
01	100	1000	0111	001	TLBI VAE2
01	100	1000	0111	100	TLBI ALLE1
01	100	1000	0111	101	TLBI VALE2
01	100	1000	0111	110	TLBI VMALLS12E1
01	100	1001	0000	001	TLBI IPAS2E1ISNXS
01	100	1001	0000	010	TLBI RIPAS2E1ISNXS
01	100	1001	0000	101	TLBI IPAS2LE1ISNXS
01	100	1001	0000	110	TLBI RIPAS2LE1ISNXS
01	100	1001	0001	000	TLBI ALLE2OSNXS
01	100	1001	0001	001	TLBI VAE2OSNXS
01	100	1001	0001	100	TLBI ALLE1OSNXS
01	100	1001	0001	101	TLBI VALE2OSNXS

op0	op1	CRn	CRm	op2	Mnemonic
01	100	1001	0001	110	TLBI VMALLS12E1OSNXS
01	100	1001	0010	001	TLBI RVAE2ISNXS
01	100	1001	0010	101	TLBI RVALE2ISNXS
01	100	1001	0011	000	TLBI ALLE2ISNXS
01	100	1001	0011	001	TLBI VAE2ISNXS
01	100	1001	0011	100	TLBI ALLE1ISNXS
01	100	1001	0011	101	TLBI VALE2ISNXS
01	100	1001	0011	110	TLBI VMALLS12E1ISNXS
01	100	1001	0100	000	TLBI IPAS2E1OSNXS
01	100	1001	0100	001	TLBI IPAS2E1NXS
01	100	1001	0100	010	TLBI RIPAS2E1NXS
01	100	1001	0100	011	TLBI RIPAS2E1OSNXS
01	100	1001	0100	100	TLBI IPAS2LE1OSNXS
01	100	1001	0100	101	TLBI IPAS2LE1NXS
01	100	1001	0100	110	TLBI RIPAS2LE1NXS
01	100	1001	0100	111	TLBI RIPAS2LE1OSNXS
01	100	1001	0101	001	TLBI RVAE2OSNXS
01	100	1001	0101	101	TLBI RVALE2OSNXS
01	100	1001	0110	001	TLBI RVAE2NXS
01	100	1001	0110	101	TLBI RVALE2NXS
01	100	1001	0111	000	TLBI ALLE2NXS
01	100	1001	0111	001	TLBI VAE2NXS
01	100	1001	0111	100	TLBI ALLE1NXS
01	100	1001	0111	101	TLBI VALE2NXS
01	100	1001	0111	110	TLBI VMALLS12E1NXS
01	110	1000	0001	000	TLBI ALLE3OS
01	110	1000	0001	001	TLBI VAE3OS
01	110	1000	0001	100	TLBI PAALLOS
01	110	1000	0001	101	TLBI VALE3OS
01	110	1000	0010	001	TLBI RVAE3IS
01	110	1000	0010	101	TLBI RVALE3IS
01	110	1000	0011	000	TLBI ALLE3IS
01	110	1000	0011	001	TLBI VAE3IS
01	110	1000	0011	101	TLBI VALE3IS
01	110	1000	0100	011	TLBI RPAOS
01	110	1000	0100	111	TLBI RPALOS
01	110	1000	0101	001	TLBI RVAE3OS
01	110	1000	0101	101	TLBI RVALE3OS
01	110	1000	0110	001	TLBI RVAE3
01	110	1000	0110	101	TLBI RVALE3
01	110	1000	0111	000	TLBI ALLE3
01	110	1000	0111	001	TLBI VAE3
01	110	1000	0111	100	TLBI PAALL
01	110	1000	0111	101	TLBI VALE3
01	110	1001	0001	000	TLBI ALLE3OSNXS
01	110	1001	0001	001	TLBI VAE3OSNXS
01	110	1001	0001	101	TLBI VALE3OSNXS
01	110	1001	0010	001	TLBI RVAE3ISNXS
01	110	1001	0010	101	TLBI RVALE3ISNXS

op0	op1	CRn	CRm	op2	Mnemonic
01	110	1001	0011	000	TLBI ALLE3ISNXS
01	110	1001	0011	001	TLBI VAE3ISNXS
01	110	1001	0011	101	TLBI VALE3ISNXS
01	110	1001	0101	001	TLBI RVAE3OSNXS
01	110	1001	0101	101	TLBI RVALE3OSNXS
01	110	1001	0110	001	TLBI RVAE3NXS
01	110	1001	0110	101	TLBI RVALE3NXS
01	110	1001	0111	000	TLBI ALLE3NXS
01	110	1001	0111	001	TLBI VAE3NXS
01	110	1001	0111	101	TLBI VALE3NXS

Accessed using TLBIP:

op0	op1	CRn	CRm	op2	Mnemonic
01	000	1000	0001	001	TLBIP VAE1OS
01	000	1000	0001	011	TLBIP VAAE1OS
01	000	1000	0001	101	TLBIP VALE1OS
01	000	1000	0001	111	TLBIP VAALE1OS
01	000	1000	0010	001	TLBIP RVAE1IS
01	000	1000	0010	011	TLBIP RVAAE1IS
01	000	1000	0010	101	TLBIP RVALE1IS
01	000	1000	0010	111	TLBIP RVAALE1IS
01	000	1000	0011	001	TLBIP VAE1IS
01	000	1000	0011	011	TLBIP VAAE1IS
01	000	1000	0011	101	TLBIP VALE1IS
01	000	1000	0011	111	TLBIP VAALE1IS
01	000	1000	0101	001	TLBIP RVAE1OS
01	000	1000	0101	011	TLBIP RVAAE1OS
01	000	1000	0101	101	TLBIP RVALE1OS
01	000	1000	0101	111	TLBIP RVAALE1OS
01	000	1000	0110	001	TLBIP RVAE1
01	000	1000	0110	011	TLBIP RVAAE1
01	000	1000	0110	101	TLBIP RVALE1
01	000	1000	0110	111	TLBIP RVAALE1
01	000	1000	0111	001	TLBIP VAE1
01	000	1000	0111	011	TLBIP VAAE1
01	000	1000	0111	101	TLBIP VALE1
01	000	1000	0111	111	TLBIP VAALE1
01	000	1001	0001	001	TLBIP VAE1OSNXS
01	000	1001	0001	011	TLBIP VAAE1OSNXS
01	000	1001	0001	101	TLBIP VALE1OSNXS
01	000	1001	0001	111	TLBIP VAALE1OSNXS
01	000	1001	0010	001	TLBIP RVAE1ISNXS
01	000	1001	0010	011	TLBIP RVAAE1ISNXS
01	000	1001	0010	101	TLBIP RVALE1ISNXS
01	000	1001	0010	111	TLBIP RVAALE1ISNXS
01	000	1001	0011	001	TLBIP VAE1ISNXS
01	000	1001	0011	011	TLBIP VAAE1ISNXS
01	000	1001	0011	101	TLBIP VALE1ISNXS

op0	op1	CRn	CRm	op2	Mnemonic
01	000	1001	0011	111	TLBIP VAALE1ISNXS
01	000	1001	0101	001	TLBIP RVAE1OSNXS
01	000	1001	0101	011	TLBIP RVAAE1OSNXS
01	000	1001	0101	101	TLBIP RVALE1OSNXS
01	000	1001	0101	111	TLBIP RVAALE1OSNXS
01	000	1001	0110	001	TLBIP RVAE1NXS
01	000	1001	0110	011	TLBIP RVAAE1NXS
01	000	1001	0110	101	TLBIP RVALE1NXS
01	000	1001	0110	111	TLBIP RVAALE1NXS
01	000	1001	0111	001	TLBIP VAE1NXS
01	000	1001	0111	011	TLBIP VAAE1NXS
01	000	1001	0111	101	TLBIP VALE1NXS
01	000	1001	0111	111	TLBIP VAALE1NXS
01	100	1000	0000	001	TLBIP IPAS2E1IS
01	100	1000	0000	010	TLBIP RIPAS2E1IS
01	100	1000	0000	101	TLBIP IPAS2LE1IS
01	100	1000	0000	110	TLBIP RIPAS2LE1IS
01	100	1000	0001	001	TLBIP VAE2OS
01	100	1000	0001	101	TLBIP VALE2OS
01	100	1000	0010	001	TLBIP RVAE2IS
01	100	1000	0010	101	TLBIP RVALE2IS
01	100	1000	0011	001	TLBIP VAE2IS
01	100	1000	0011	101	TLBIP VALE2IS
01	100	1000	0100	000	TLBIP IPAS2E1OS
01	100	1000	0100	001	TLBIP IPAS2E1
01	100	1000	0100	010	TLBIP RIPAS2E1
01	100	1000	0100	011	TLBIP RIPAS2E1OS
01	100	1000	0100	100	TLBIP IPAS2LE1OS
01	100	1000	0100	101	TLBIP IPAS2LE1
01	100	1000	0100	110	TLBIP RIPAS2LE1
01	100	1000	0100	111	TLBIP RIPAS2LE1OS
01	100	1000	0101	001	TLBIP RVAE2OS
01	100	1000	0101	101	TLBIP RVALE2OS
01	100	1000	0110	001	TLBIP RVAE2
01	100	1000	0110	101	TLBIP RVALE2
01	100	1000	0111	001	TLBIP VAE2
01	100	1000	0111	101	TLBIP VALE2
01	100	1001	0000	001	TLBIP IPAS2E1ISNXS
01	100	1001	0000	010	TLBIP RIPAS2E1ISNXS
01	100	1001	0000	101	TLBIP IPAS2LE1ISNXS
01	100	1001	0000	110	TLBIP RIPAS2LE1ISNXS
01	100	1001	0001	001	TLBIP VAE2OSNXS
01	100	1001	0001	101	TLBIP VALE2OSNXS
01	100	1001	0010	001	TLBIP RVAE2ISNXS
01	100	1001	0010	101	TLBIP RVALE2ISNXS
01	100	1001	0011	001	TLBIP VAE2ISNXS
01	100	1001	0011	101	TLBIP VALE2ISNXS
01	100	1001	0100	000	TLBIP IPAS2E1OSNXS
01	100	1001	0100	001	TLBIP IPAS2E1NXS

op0	op1	CRn	CRm	op2	Mnemonic
01	100	1001	0100	010	TLBIP RIPAS2E1NXS
01	100	1001	0100	011	TLBIP RIPAS2E1OSNXS
01	100	1001	0100	100	TLBIP IPAS2LE1OSNXS
01	100	1001	0100	101	TLBIP IPAS2LE1NXS
01	100	1001	0100	110	TLBIP RIPAS2LE1NXS
01	100	1001	0100	111	TLBIP RIPAS2LE1OSNXS
01	100	1001	0101	001	TLBIP RVAE2OSNXS
01	100	1001	0101	101	TLBIP RVAE2OSNXS
01	100	1001	0110	001	TLBIP RVAE2NXS
01	100	1001	0110	101	TLBIP RVAE2NXS
01	100	1001	0111	001	TLBIP VAE2NXS
01	100	1001	0111	101	TLBIP VAE2NXS
01	110	1000	0001	001	TLBIP VAE3OS
01	110	1000	0001	101	TLBIP VAE3OS
01	110	1000	0010	001	TLBIP RVAE3IS
01	110	1000	0010	101	TLBIP RVAE3IS
01	110	1000	0011	001	TLBIP VAE3IS
01	110	1000	0011	101	TLBIP VAE3IS
01	110	1000	0101	001	TLBIP RVAE3OS
01	110	1000	0101	101	TLBIP RVAE3OS
01	110	1000	0110	001	TLBIP RVAE3
01	110	1000	0110	101	TLBIP RVAE3
01	110	1000	0111	001	TLBIP VAE3
01	110	1000	0111	101	TLBIP VAE3
01	110	1001	0001	001	TLBIP VAE3OSNXS
01	110	1001	0001	101	TLBIP VAE3OSNXS
01	110	1001	0010	001	TLBIP RVAE3ISNXS
01	110	1001	0010	101	TLBIP RVAE3ISNXS
01	110	1001	0011	001	TLBIP VAE3ISNXS
01	110	1001	0011	101	TLBIP VAE3ISNXS
01	110	1001	0101	001	TLBIP RVAE3OSNXS
01	110	1001	0101	101	TLBIP RVAE3OSNXS
01	110	1001	0110	001	TLBIP RVAE3NXS
01	110	1001	0110	101	TLBIP RVAE3NXS
01	110	1001	0111	001	TLBIP VAE3NXS
01	110	1001	0111	101	TLBIP VAE3NXS

3005/0907/2022 17:16:03

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

IndexSystem Register index by functional group

Below are indexes for registers with the following main functional groups:

- [ID](#)
- [Memory](#)
- [Other](#)
- [Exception](#)
- [Special](#)
- [PSTATE](#)
- [Cache](#)
- [Address](#)
- [TLB](#)
- [PMU](#)
- [Reset](#)
- [Thread](#)
- [IMP DEF](#)
- [Timer](#)
- [Debug](#)
- [CTI](#)
- [Virt](#)
- [Secure](#)
- [Float](#)
- [Legacy](#)
- [Trace](#)
- [GIC](#)
- [GICD](#)
- [GICR](#)
- [GICC](#)
- [GICV](#)
- [GICH](#)
- [GITS](#)
- [RAS](#)
- [MPAM](#)
- [Pointer authentication](#)
- [AMU](#)
- [Root](#)
- [GIC ITS registers](#)

In the ID functional group:

Exec state	Name	Description
AArch32	CCSIDR	Current Cache Size ID Register
AArch32	CCSIDR2	Current Cache Size ID Register 2
AArch32	CLIDR	Cache Level ID Register
AArch32	CSSELR	Cache Size Selection Register
AArch32	CTR	Cache Type Register
AArch32	ID_AFR0	Auxiliary Feature Register 0
AArch32	ID_DFR0	Debug Feature Register 0
AArch32	ID_DFR1	Debug Feature Register 1
AArch32	ID_ISAR0	Instruction Set Attribute Register 0
AArch32	ID_ISAR1	Instruction Set Attribute Register 1
AArch32	ID_ISAR2	Instruction Set Attribute Register 2
AArch32	ID_ISAR3	Instruction Set Attribute Register 3
AArch32	ID_ISAR4	Instruction Set Attribute Register 4
AArch32	ID_ISAR5	Instruction Set Attribute Register 5
AArch32	ID_ISAR6	Instruction Set Attribute Register 6
AArch32	ID_MMFR0	Memory Model Feature Register 0
AArch32	ID_MMFR1	Memory Model Feature Register 1
AArch32	ID_MMFR2	Memory Model Feature Register 2
AArch32	ID_MMFR3	Memory Model Feature Register 3
AArch32	ID_MMFR4	Memory Model Feature Register 4

Exec state	Name	Description
AArch32	ID_MMFR5	Memory Model Feature Register 5
AArch32	ID_PFR0	Processor Feature Register 0
AArch32	ID_PFR1	Processor Feature Register 1
AArch32	ID_PFR2	Processor Feature Register 2
AArch32	MIDR	Main ID Register
AArch32	MPIDR	Multiprocessor Affinity Register
AArch32	REVIDR	Revision ID Register
AArch32	TCMTR	TCM Type Register
AArch32	TLBTR	TLB Type Register
AArch64	CCSIDR2_EL1	Current Cache Size ID Register 2
AArch64	CCSIDR_EL1	Current Cache Size ID Register
AArch64	CLIDR_EL1	Cache Level ID Register
AArch64	CSSELR_EL1	Cache Size Selection Register
AArch64	CTR_EL0	Cache Type Register
AArch64	DCZID_EL0	Data Cache Zero ID register
AArch64	GMID_EL1	Multiple tag transfer ID register
AArch64	ID_AA64AFR0_EL1	AArch64 Auxiliary Feature Register 0
AArch64	ID_AA64AFR1_EL1	AArch64 Auxiliary Feature Register 1
AArch64	ID_AA64DFR0_EL1	AArch64 Debug Feature Register 0
AArch64	ID_AA64DFR1_EL1	AArch64 Debug Feature Register 1
AArch64	ID_AA64ISAR0_EL1	AArch64 Instruction Set Attribute Register 0
AArch64	ID_AA64ISAR1_EL1	AArch64 Instruction Set Attribute Register 1
AArch64	ID_AA64ISAR2_EL1	AArch64 Instruction Set Attribute Register 2
AArch64	ID_AA64MMFR0_EL1	AArch64 Memory Model Feature Register 0
AArch64	ID_AA64MMFR1_EL1	AArch64 Memory Model Feature Register 1
AArch64	ID_AA64MMFR2_EL1	AArch64 Memory Model Feature Register 2
AArch64	ID_AA64MMFR3_EL1	AArch64 Memory Model Feature Register 3
AArch64	ID_AA64MMFR4_EL1	AArch64 Memory Model Feature Register 4
AArch64	ID_AA64PFR0_EL1	AArch64 Processor Feature Register 0
AArch64	ID_AA64PFR1_EL1	AArch64 Processor Feature Register 1
AArch64	ID_AA64PFR2_EL1	AArch64 Processor Feature Register 2
AArch64	ID_AA64SMFR0_EL1	SME Feature ID register 0
AArch64	ID_AA64ZFR0_EL1	SVE Feature ID register 0
AArch64	ID_AFR0_EL1	AArch32 Auxiliary Feature Register 0
AArch64	ID_DFR0_EL1	AArch32 Debug Feature Register 0
AArch64	ID_DFR1_EL1	Debug Feature Register 1
AArch64	ID_ISAR0_EL1	AArch32 Instruction Set Attribute Register 0
AArch64	ID_ISAR1_EL1	AArch32 Instruction Set Attribute Register 1
AArch64	ID_ISAR2_EL1	AArch32 Instruction Set Attribute Register 2
AArch64	ID_ISAR3_EL1	AArch32 Instruction Set Attribute Register 3
AArch64	ID_ISAR4_EL1	AArch32 Instruction Set Attribute Register 4
AArch64	ID_ISAR5_EL1	AArch32 Instruction Set Attribute Register 5
AArch64	ID_ISAR6_EL1	AArch32 Instruction Set Attribute Register 6
AArch64	ID_MMFR0_EL1	AArch32 Memory Model Feature Register 0
AArch64	ID_MMFR1_EL1	AArch32 Memory Model Feature Register 1
AArch64	ID_MMFR2_EL1	AArch32 Memory Model Feature Register 2
AArch64	ID_MMFR3_EL1	AArch32 Memory Model Feature Register 3
AArch64	ID_MMFR4_EL1	AArch32 Memory Model Feature Register 4
AArch64	ID_MMFR5_EL1	AArch32 Memory Model Feature Register 5
AArch64	ID_PFR0_EL1	AArch32 Processor Feature Register 0
AArch64	ID_PFR1_EL1	AArch32 Processor Feature Register 1
AArch64	ID_PFR2_EL1	AArch32 Processor Feature Register 2
AArch64	MIDR_EL1	Main ID Register
AArch64	MPAMIDR_EL1	MPAM ID Register (EL1)
AArch64	MPIDR_EL1	Multiprocessor Affinity Register
AArch64	REVIDR_EL1	Revision ID Register
AArch64	SMIDR_EL1	Streaming Mode Identification Register
External	EDAA32PFR	External Debug Auxiliary Processor Feature Register
External	EDDFR	External Debug Feature Register
External	EDPFR	External Debug Processor Feature Register
External	MIDR_EL1	Main ID Register

In the Memory functional group:

Exec state	Name	Description
AArch32	AMAIRO	Auxiliary Memory Attribute Indirection Register 0
AArch32	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
AArch32	CONTEXTIDR	Context ID Register
AArch32	DACR	Domain Access Control Register
AArch32	HAMAIRO	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch32	HMAIRO	Hyp Memory Attribute Indirection Register 0
AArch32	HMAIR1	Hyp Memory Attribute Indirection Register 1
AArch32	HTCR	Hyp Translation Control Register
AArch32	HTTBR	Hyp Translation Table Base Register
AArch32	MAIRO	Memory Attribute Indirection Register 0
AArch32	MAIR1	Memory Attribute Indirection Register 1
AArch32	NMRR	Normal Memory Remap Register
AArch32	PRRR	Primary Region Remap Register
AArch32	TTBCR	Translation Table Base Control Register
AArch32	TTBCR2	Translation Table Base Control Register 2
AArch32	TTBR0	Translation Table Base Register 0
AArch32	TTBR1	Translation Table Base Register 1
AArch32	VTCT	Virtualization Translation Control Register
AArch32	VTTBR	Virtualization Translation Table Base Register
AArch64	AMAIR2_EL1	Extended Auxiliary Memory Attribute Indirection Register (EL1)
AArch64	AMAIR2_EL2	Extended Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	AMAIR2_EL3	Extended Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	AMAIR_EL1	Auxiliary Memory Attribute Indirection Register (EL1)
AArch64	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	AMAIR_EL3	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	CONTEXTIDR_EL1	Context ID Register (EL1)
AArch64	CONTEXTIDR_EL2	Context ID Register (EL2)
AArch64	DACR32_EL2	Domain Access Control Register
AArch64	GPCCR_EL3	Granule Protection Check Control Register (EL3)
AArch64	GPTBR_EL3	Granule Protection Table Base Register
AArch64	LORC_EL1	LORegion Control (EL1)
AArch64	LOREA_EL1	LORegion End Address (EL1)
AArch64	LORID_EL1	LORegionID (EL1)
AArch64	LORN_EL1	LORegion Number (EL1)
AArch64	LORSA_EL1	LORegion Start Address (EL1)
AArch64	MAIR2_EL1	Extended Memory Attribute Indirection Register (EL1)
AArch64	MAIR2_EL2	Extended Memory Attribute Indirection Register (EL2)
AArch64	MAIR2_EL3	Extended Memory Attribute Indirection Register (EL3)
AArch64	MAIR_EL1	Memory Attribute Indirection Register (EL1)
AArch64	MAIR_EL2	Memory Attribute Indirection Register (EL2)
AArch64	MAIR_EL3	Memory Attribute Indirection Register (EL3)
AArch64	PIRE0_EL1	Permission Indirection Register 0 (EL1)
AArch64	PIRE0_EL2	Permission Indirection Register 0 (EL2)
AArch64	PIR_EL1	Permission Indirection Register 1 (EL1)
AArch64	PIR_EL2	Permission Indirection Register 2 (EL2)
AArch64	PIR_EL3	Permission Indirection Register 3 (EL3)
AArch64	POR_EL0	Permission Overlay Register 0 (EL0)
AArch64	POR_EL1	Permission Overlay Register 1 (EL1)
AArch64	POR_EL2	Permission Overlay Register 2 (EL2)
AArch64	POR_EL3	Permission Overlay Register 3 (EL3)
AArch64	RCWMASK_EL1	Read Check Write Instruction Mask (EL1)
AArch64	RCWSMASK_EL1	Software Read Check Write Instruction Mask (EL1)
AArch64	S2PIR_EL2	Stage 2 Permission Indirection Register (EL2)
AArch64	S2POR_EL1	Stage 2 Permission Overlay Register (EL1)
AArch64	TCR2_EL1	Extended Translation Control Register (EL1)
AArch64	TCR2_EL2	Extended Translation Control Register (EL2)
AArch64	TCR_EL1	Translation Control Register (EL1)
AArch64	TCR_EL2	Translation Control Register (EL2)
AArch64	TCR_EL3	Translation Control Register (EL3)
AArch64	TTBR0_EL1	Translation Table Base Register 0 (EL1)
AArch64	TTBR0_EL2	Translation Table Base Register 0 (EL2)

Exec state	Name	Description
AArch64	TTBR0_EL3	Translation Table Base Register 0 (EL3)
AArch64	TTBR1_EL1	Translation Table Base Register 1 (EL1)
AArch64	TTBR1_EL2	Translation Table Base Register 1 (EL2)
AArch64	VTCR_EL2	Virtualization Translation Control Register
AArch64	VTTBR_EL2	Virtualization Translation Table Base Register

In the Other functional group:

Exec state	Name	Description
AArch32	CPACR	Architectural Feature Access Control Register
AArch32	SCTLR	System Control Register
AArch64	CPACR_EL1	Architectural Feature Access Control Register
AArch64	SCTLR2_EL1	System Control Register (EL1)
AArch64	SCTLR2_EL3	System Control Register (EL3)
AArch64	SCTLR_EL1	System Control Register (EL1)
AArch64	SCTLR_EL3	System Control Register (EL3)
AArch64	SMCR_EL1	SME Control Register (EL1)
AArch64	SMCR_EL2	SME Control Register (EL2)
AArch64	SMCR_EL3	SME Control Register (EL3)
AArch64	SMPRIMAP_EL2	Streaming Mode Priority Mapping Register
AArch64	SMPRI_EL1	Streaming Mode Priority Register
AArch64	ZCR_EL1	SVE Control Register (EL1)
AArch64	ZCR_EL2	SVE Control Register (EL2)
AArch64	ZCR_EL3	SVE Control Register (EL3)

In the Exception functional group:

Exec state	Name	Description
AArch32	ADFSR	Auxiliary Data Fault Status Register
AArch32	AIFSR	Auxiliary Instruction Fault Status Register
AArch32	DFAR	Data Fault Address Register
AArch32	DFSR	Data Fault Status Register
AArch32	HADFSR	Hyp Auxiliary Data Fault Status Register
AArch32	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
AArch32	HDFAR	Hyp Data Fault Address Register
AArch32	HIFAR	Hyp Instruction Fault Address Register
AArch32	HPFAR	Hyp IPA Fault Address Register
AArch32	HSR	Hyp Syndrome Register
AArch32	HVBAR	Hyp Vector Base Address Register
AArch32	IFAR	Instruction Fault Address Register
AArch32	IFSR	Instruction Fault Status Register
AArch32	ISR	Interrupt Status Register
AArch32	MVBAR	Monitor Vector Base Address Register
AArch32	VBAR	Vector Base Address Register
AArch64	AFSR0_EL1	Auxiliary Fault Status Register 0 (EL1)
AArch64	AFSR0_EL2	Auxiliary Fault Status Register 0 (EL2)
AArch64	AFSR0_EL3	Auxiliary Fault Status Register 0 (EL3)
AArch64	AFSR1_EL1	Auxiliary Fault Status Register 1 (EL1)
AArch64	AFSR1_EL2	Auxiliary Fault Status Register 1 (EL2)
AArch64	AFSR1_EL3	Auxiliary Fault Status Register 1 (EL3)
AArch64	ESR_EL1	Exception Syndrome Register (EL1)
AArch64	ESR_EL2	Exception Syndrome Register (EL2)
AArch64	ESR_EL3	Exception Syndrome Register (EL3)
AArch64	FAR_EL1	Fault Address Register (EL1)
AArch64	FAR_EL2	Fault Address Register (EL2)
AArch64	FAR_EL3	Fault Address Register (EL3)
AArch64	HPFAR_EL2	Hypervisor IPA Fault Address Register
AArch64	IFSR32_EL2	Instruction Fault Status Register (EL2)
AArch64	ISR_EL1	Interrupt Status Register
AArch64	MFAR_EL3	Physical PA Fault Address Register (EL3)
AArch64	VBAR_EL1	Vector Base Address Register (EL1)
AArch64	VBAR_EL2	Vector Base Address Register (EL2)

Exec state	Name	Description
AArch64	VBAR_EL3	Vector Base Address Register (EL3)

In the Special functional group:

Exec state	Name	Description
AArch32	DLR	Debug Link Register
AArch32	DSPSR	Debug Saved Program Status Register
AArch32	ELR_hyp	Exception Link Register (Hyp mode)
AArch32	SPSR	Saved Program Status Register
AArch32	SPSR_abt	Saved Program Status Register (Abort mode)
AArch32	SPSR_fiq	Saved Program Status Register (FIQ mode)
AArch32	SPSR_hyp	Saved Program Status Register (Hyp mode)
AArch32	SPSR_irq	Saved Program Status Register (IRQ mode)
AArch32	SPSR_mon	Saved Program Status Register (Monitor mode)
AArch32	SPSR_svc	Saved Program Status Register (Supervisor mode)
AArch32	SPSR_und	Saved Program Status Register (Undefined mode)
AArch64	ELR_EL1	Exception Link Register (EL1)
AArch64	ELR_EL2	Exception Link Register (EL2)
AArch64	ELR_EL3	Exception Link Register (EL3)
AArch64	SPSR_EL1	Saved Program Status Register (EL1)
AArch64	SPSR_EL2	Saved Program Status Register (EL2)
AArch64	SPSR_EL3	Saved Program Status Register (EL3)
AArch64	SPSR_abt	Saved Program Status Register (Abort mode)
AArch64	SPSR_fiq	Saved Program Status Register (FIQ mode)
AArch64	SPSR_irq	Saved Program Status Register (IRQ mode)
AArch64	SPSR_und	Saved Program Status Register (Undefined mode)
AArch64	SP_EL0	Stack Pointer (EL0)
AArch64	SP_EL1	Stack Pointer (EL1)
AArch64	SP_EL2	Stack Pointer (EL2)
AArch64	SP_EL3	Stack Pointer (EL3)

In the PSTATE functional group:

Exec state	Name	Description
AArch32	APSR	Application Program Status Register
AArch32	CPSR	Current Program Status Register
AArch64	ALLINT	All Interrupt Mask Bit
AArch64	CurrentEL	Current Exception Level
AArch64	DAIF	Interrupt Mask Bits
AArch64	DIT	Data Independent Timing
AArch64	NZCV	Condition Flags
AArch64	PAN	Privileged Access Never
AArch64	PM	PMU Exception Mask
AArch64	SPSel	Stack Pointer Select
AArch64	SSBS	Speculative Store Bypass Safe
AArch64	SVCR	Streaming Vector Control Register
AArch64	TCO	Tag Check Override
AArch64	UAO	User Access Override

In the Cache functional group:

Exec state	Name	Description
AArch32	BPIALL	Branch Predictor Invalidate All
AArch32	BPIALLIS	Branch Predictor Invalidate All, Inner Shareable
AArch32	BPIMVA	Branch Predictor Invalidate by VA
AArch32	DCCIMVAC	Data Cache line Clean and Invalidate by VA to PoC
AArch32	DCCISW	Data Cache line Clean and Invalidate by Set/Way
AArch32	DCCMVAC	Data Cache line Clean by VA to PoC
AArch32	DCCMVAU	Data Cache line Clean by VA to PoU
AArch32	DCCSW	Data Cache line Clean by Set/Way
AArch32	DCIMVAC	Data Cache line Invalidate by VA to PoC

Exec state	Name	Description
AArch32	DCISW	Data Cache line Invalidate by Set/Way
AArch32	ICIALLU	Instruction Cache Invalidate All to PoU
AArch32	ICIALLUIS	Instruction Cache Invalidate All to PoU, Inner Shareable
AArch32	ICIMVAU	Instruction Cache line Invalidate by VA to PoU
AArch64	DC CGDSW	Clean of Data and Allocation Tags by Set/Way
AArch64	DC CGDVAC	Clean of Data and Allocation Tags by VA to PoC
AArch64	DC CGDVADP	Clean of Data and Allocation Tags by VA to PoDP
AArch64	DC CGDVAP	Clean of Data and Allocation Tags by VA to PoP
AArch64	DC CGSW	Clean of Allocation Tags by Set/Way
AArch64	DC CGVAC	Clean of Allocation Tags by VA to PoC
AArch64	DC CGVADP	Clean of Allocation Tags by VA to PoDP
AArch64	DC CGVAP	Clean of Allocation Tags by VA to PoP
AArch64	DC CIGDPAE	Clean and invalidate of data and allocation tags by PA to PoE
AArch64	DC CIGDPAPA	Clean and Invalidate of Data and Allocation Tags by PA to PoPA
AArch64	DC CIGDSW	Clean and Invalidate of Data and Allocation Tags by Set/Way
AArch64	DC CIGDVAC	Clean and Invalidate of Data and Allocation Tags by VA to PoC
AArch64	DC CIGSW	Clean and Invalidate of Allocation Tags by Set/Way
AArch64	DC CIGVAC	Clean and Invalidate of Allocation Tags by VA to PoC
AArch64	DC CIPAE	Data or unified Cache line Clean and Invalidate by PA to PoE
AArch64	DC CIPAPA	Data or unified Cache line Clean and Invalidate by PA to PoPA
AArch64	DC CISW	Data or unified Cache line Clean and Invalidate by Set/Way
AArch64	DC CIVAC	Data or unified Cache line Clean and Invalidate by VA to PoC
AArch64	DC CSW	Data or unified Cache line Clean by Set/Way
AArch64	DC CVAC	Data or unified Cache line Clean by VA to PoC
AArch64	DC CVADP	Data or unified Cache line Clean by VA to PoDP
AArch64	DC CVAP	Data or unified Cache line Clean by VA to PoP
AArch64	DC CVAU	Data or unified Cache line Clean by VA to PoU
AArch64	DC GVA	Data Cache set Allocation Tag by VA
AArch64	DC GZVA	Data Cache set Allocation Tags and Zero by VA
AArch64	DC IGDSW	Invalidate of Data and Allocation Tags by Set/Way
AArch64	DC IGDVAC	Invalidate of Data and Allocation Tags by VA to PoC
AArch64	DC IGSW	Invalidate of Allocation Tags by Set/Way
AArch64	DC IGVAC	Invalidate of Allocation Tags by VA to PoC
AArch64	DC ISW	Data or unified Cache line Invalidate by Set/Way
AArch64	DC IVAC	Data or unified Cache line Invalidate by VA to PoC
AArch64	DC ZVA	Data Cache Zero by VA
AArch64	IC IALLU	Instruction Cache Invalidate All to PoU
AArch64	IC IALLUIS	Instruction Cache Invalidate All to PoU, Inner Shareable
AArch64	IC IVAU	Instruction Cache line Invalidate by VA to PoU

In the Address functional group:

Exec state	Name	Description
AArch32	ATS12NSOPR	Address Translate Stages 1 and 2 Non-secure Only PL1 Read
AArch32	ATS12NSOPW	Address Translate Stages 1 and 2 Non-secure Only PL1 Write
AArch32	ATS12NSOUR	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read
AArch32	ATS12NSOUW	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write
AArch32	ATS1CPR	Address Translate Stage 1 Current state PL1 Read
AArch32	ATS1CPRP	Address Translate Stage 1 Current state PL1 Read PAN
AArch32	ATS1CPW	Address Translate Stage 1 Current state PL1 Write
AArch32	ATS1CPWP	Address Translate Stage 1 Current state PL1 Write PAN
AArch32	ATS1CUR	Address Translate Stage 1 Current state Unprivileged Read
AArch32	ATS1CUW	Address Translate Stage 1 Current state Unprivileged Write
AArch32	ATS1HR	Address Translate Stage 1 Hyp mode Read
AArch32	ATS1HW	Address Translate Stage 1 Hyp mode Write
AArch32	PAR	Physical Address Register
AArch64	AT S12E0R	Address Translate Stages 1 and 2 EL0 Read
AArch64	AT S12E0W	Address Translate Stages 1 and 2 EL0 Write
AArch64	AT S12E1R	Address Translate Stages 1 and 2 EL1 Read
AArch64	AT S12E1W	Address Translate Stages 1 and 2 EL1 Write
AArch64	AT S1E0R	Address Translate Stage 1 EL0 Read
AArch64	AT S1E0W	Address Translate Stage 1 EL0 Write
AArch64	AT S1E1R	Address Translate Stage 1 EL1 Read

Exec state	Name	Description
AArch64	AT S1E1RP	Address Translate Stage 1 EL1 Read PAN
AArch64	AT S1E1W	Address Translate Stage 1 EL1 Write
AArch64	AT S1E1WP	Address Translate Stage 1 EL1 Write PAN
AArch64	AT S1E2R	Address Translate Stage 1 EL2 Read
AArch64	AT S1E2W	Address Translate Stage 1 EL2 Write
AArch64	AT S1E3R	Address Translate Stage 1 EL3 Read
AArch64	AT S1E3W	Address Translate Stage 1 EL3 Write
AArch64	PAR_EL1	Physical Address Register

In the TLB functional group:

Exec state	Name	Description
AArch32	CFPRCTX	Control Flow Prediction Restriction by Context
AArch32	COSPRCTX	Clear Oher Speculative Restriction by Context
AArch32	CPPRCTX	Cache Prefetch Prediction Restriction by Context
AArch32	DTLBIALL	Data TLB Invalidate All
AArch32	DTLBIASID	Data TLB Invalidate by ASID match
AArch32	DTLBIMVA	Data TLB Invalidate by VA
AArch32	DVPRCTX	Data Value Prediction Restriction by Context
AArch32	ITLBIALL	Instruction TLB Invalidate All
AArch32	ITLBIASID	Instruction TLB Invalidate by ASID match
AArch32	ITLBIMVA	Instruction TLB Invalidate by VA
AArch32	TLBIALL	TLB Invalidate All
AArch32	TLBIALLH	TLB Invalidate All, Hyp mode
AArch32	TLBIALLHIS	TLB Invalidate All, Hyp mode, Inner Shareable
AArch32	TLBIALLIS	TLB Invalidate All, Inner Shareable
AArch32	TLBIALLNSNH	TLB Invalidate All, Non-Secure Non-Hyp
AArch32	TLBIALLNSNHIS	TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable
AArch32	TLBIASID	TLB Invalidate by ASID match
AArch32	TLBIASIDIS	TLB Invalidate by ASID match, Inner Shareable
AArch32	TLBIIPAS2	TLB Invalidate by Intermediate Physical Address, Stage 2
AArch32	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
AArch32	TLBIIPAS2L	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
AArch32	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
AArch32	TLBIMVA	TLB Invalidate by VA
AArch32	TLBIMVAA	TLB Invalidate by VA, All ASID
AArch32	TLBIMVAAIS	TLB Invalidate by VA, All ASID, Inner Shareable
AArch32	TLBIMVAAL	TLB Invalidate by VA, All ASID, Last level
AArch32	TLBIMVAALIS	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
AArch32	TLBIMVAH	TLB Invalidate by VA, Hyp mode
AArch32	TLBIMVAHIS	TLB Invalidate by VA, Hyp mode, Inner Shareable
AArch32	TLBIMVAIS	TLB Invalidate by VA, Inner Shareable
AArch32	TLBIMVAL	TLB Invalidate by VA, Last level
AArch32	TLBIMVALH	TLB Invalidate by VA, Last level, Hyp mode
AArch32	TLBIMVALHIS	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
AArch32	TLBIMVALIS	TLB Invalidate by VA, Last level, Inner Shareable
AArch64	TLBI ALLE1, TLBI ALLE1NXS	TLB Invalidate All, EL1
AArch64	TLBI ALLE1IS, TLBI ALLE1ISNXS	TLB Invalidate All, EL1, Inner Shareable
AArch64	TLBI ALLE1OS, TLBI ALLE1OSNXS	TLB Invalidate All, EL1, Outer Shareable
AArch64	TLBI ALLE2, TLBI ALLE2NXS	TLB Invalidate All, EL2
AArch64	TLBI ALLE2IS, TLBI ALLE2ISNXS	TLB Invalidate All, EL2, Inner Shareable
AArch64	TLBI ALLE2OS, TLBI ALLE2OSNXS	TLB Invalidate All, EL2, Outer Shareable
AArch64	TLBI ALLE3, TLBI ALLE3NXS	TLB Invalidate All, EL3
AArch64	TLBI ALLE3IS, TLBI ALLE3ISNXS	TLB Invalidate All, EL3, Inner Shareable

Exec state	Name	Description
AArch64	TLBI ALLE3OS, TLBI ALLE3OSNXS	TLB Invalidate All, EL3, Outer Shareable
AArch64	TLBI ASIDE1, TLBI ASIDE1NXS	TLB Invalidate by ASID, EL1
AArch64	TLBI ASIDE1IS, TLBI ASIDE1ISNXS	TLB Invalidate by ASID, EL1, Inner Shareable
AArch64	TLBI ASIDE1OS, TLBI ASIDE1OSNXS	TLB Invalidate by ASID, EL1, Outer Shareable
AArch64	TLBI IPAS2E1, TLBI IPAS2E1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBI IPAS2E1IS, TLBI IPAS2E1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBI IPAS2E1OS, TLBI IPAS2E1OSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBI IPAS2LE1, TLBI IPAS2LE1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBI IPAS2LE1IS, TLBI IPAS2LE1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBI IPAS2LE1OS, TLBI IPAS2LE1OSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBI PAALL	TLB Invalidate GPT Information by PA, All Entries, Local
AArch64	TLBI PAALLOS	TLB Invalidate GPT Information by PA, All Entries, Outer Shareable
AArch64	TLBI RIPAS2E1, TLBI RIPAS2E1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBI RIPAS2E1IS, TLBI RIPAS2E1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBI RIPAS2E1OS, TLBI RIPAS2E1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBI RIPAS2LE1, TLBI RIPAS2LE1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBI RPALOS	TLB Range Invalidate GPT Information by PA, Last level, Outer Shareable
AArch64	TLBI RPAOS	TLB Range Invalidate GPT Information by PA, Outer Shareable
AArch64	TLBI RVAAE1, TLBI RVAAE1NXS	TLB Range Invalidate by VA, All ASID, EL1
AArch64	TLBI RVAAE1IS, TLBI RVAAE1ISNXS	TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable
AArch64	TLBI RVAAE1OS, TLBI RVAAE1OSNXS	TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable
AArch64	TLBI RVAALE1, TLBI RVAALE1NXS	TLB Range Invalidate by VA, All ASID, Last level, EL1
AArch64	TLBI RVAALE1IS, TLBI RVAALE1ISNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
AArch64	TLBI RVAALE1OS, TLBI RVAALE1OSNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
AArch64	TLBI RVAE1, TLBI RVAE1NXS	TLB Range Invalidate by VA, EL1
AArch64	TLBI RVAE1IS, TLBI RVAE1ISNXS	TLB Range Invalidate by VA, EL1, Inner Shareable
AArch64	TLBI RVAE1OS, TLBI RVAE1OSNXS	TLB Range Invalidate by VA, EL1, Outer Shareable
AArch64	TLBI RVAE2, TLBI RVAE2NXS	TLB Range Invalidate by VA, EL2
AArch64	TLBI RVAE2IS, TLBI RVAE2ISNXS	TLB Range Invalidate by VA, EL2, Inner Shareable
AArch64	TLBI RVAE2OS, TLBI RVAE2OSNXS	TLB Range Invalidate by VA, EL2, Outer Shareable
AArch64	TLBI RVAE3, TLBI RVAE3NXS	TLB Range Invalidate by VA, EL3
AArch64	TLBI RVAE3IS, TLBI RVAE3ISNXS	TLB Range Invalidate by VA, EL3, Inner Shareable
AArch64	TLBI RVAE3OS, TLBI RVAE3OSNXS	TLB Range Invalidate by VA, EL3, Outer Shareable
AArch64	TLBI RVALE1, TLBI RVALE1NXS	TLB Range Invalidate by VA, Last level, EL1

Exec state	Name	Description
AArch64	TLBI RVALE1IS, TLBI RVALE1ISNXS	TLB Range Invalidate by VA, Last level, EL1, Inner Shareable
AArch64	TLBI RVALE1OS, TLBI RVALE1OSNXS	TLB Range Invalidate by VA, Last level, EL1, Outer Shareable
AArch64	TLBI RVALE2, TLBI RVALE2NXS	TLB Range Invalidate by VA, Last level, EL2
AArch64	TLBI RVALE2IS, TLBI RVALE2ISNXS	TLB Range Invalidate by VA, Last level, EL2, Inner Shareable
AArch64	TLBI RVALE2OS, TLBI RVALE2OSNXS	TLB Range Invalidate by VA, Last level, EL2, Outer Shareable
AArch64	TLBI RVALE3, TLBI RVALE3NXS	TLB Range Invalidate by VA, Last level, EL3
AArch64	TLBI RVALE3IS, TLBI RVALE3ISNXS	TLB Range Invalidate by VA, Last level, EL3, Inner Shareable
AArch64	TLBI RVALE3OS, TLBI RVALE3OSNXS	TLB Range Invalidate by VA, Last level, EL3, Outer Shareable
AArch64	TLBI VAAE1, TLBI VAAE1NXS	TLB Invalidate by VA, All ASID, EL1
AArch64	TLBI VAAE1IS, TLBI VAAE1ISNXS	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
AArch64	TLBI VAAE1OS, TLBI VAAE1OSNXS	TLB Invalidate by VA, All ASID, EL1, Outer Shareable
AArch64	TLBI VAALE1, TLBI VAALE1NXS	TLB Invalidate by VA, All ASID, Last level, EL1
AArch64	TLBI VAALE1IS, TLBI VAALE1ISNXS	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
AArch64	TLBI VAALE1OS, TLBI VAALE1OSNXS	TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
AArch64	TLBI VAE1, TLBI VAE1NXS	TLB Invalidate by VA, EL1
AArch64	TLBI VAE1IS, TLBI VAE1ISNXS	TLB Invalidate by VA, EL1, Inner Shareable
AArch64	TLBI VAE1OS, TLBI VAE1OSNXS	TLB Invalidate by VA, EL1, Outer Shareable
AArch64	TLBI VAE2, TLBI VAE2NXS	TLB Invalidate by VA, EL2
AArch64	TLBI VAE2IS, TLBI VAE2ISNXS	TLB Invalidate by VA, EL2, Inner Shareable
AArch64	TLBI VAE2OS, TLBI VAE2OSNXS	TLB Invalidate by VA, EL2, Outer Shareable
AArch64	TLBI VAE3, TLBI VAE3NXS	TLB Invalidate by VA, EL3
AArch64	TLBI VAE3IS, TLBI VAE3ISNXS	TLB Invalidate by VA, EL3, Inner Shareable
AArch64	TLBI VAE3OS, TLBI VAE3OSNXS	TLB Invalidate by VA, EL3, Outer Shareable
AArch64	TLBI VALE1, TLBI VALE1NXS	TLB Invalidate by VA, Last level, EL1
AArch64	TLBI VALE1IS, TLBI VALE1ISNXS	TLB Invalidate by VA, Last level, EL1, Inner Shareable
AArch64	TLBI VALE1OS, TLBI VALE1OSNXS	TLB Invalidate by VA, Last level, EL1, Outer Shareable
AArch64	TLBI VALE2, TLBI VALE2NXS	TLB Invalidate by VA, Last level, EL2
AArch64	TLBI VALE2IS, TLBI VALE2ISNXS	TLB Invalidate by VA, Last level, EL2, Inner Shareable
AArch64	TLBI VALE2OS, TLBI VALE2OSNXS	TLB Invalidate by VA, Last level, EL2, Outer Shareable
AArch64	TLBI VALE3, TLBI VALE3NXS	TLB Invalidate by VA, Last level, EL3
AArch64	TLBI VALE3IS, TLBI VALE3ISNXS	TLB Invalidate by VA, Last level, EL3, Inner Shareable
AArch64	TLBI VALE3OS, TLBI VALE3OSNXS	TLB Invalidate by VA, Last level, EL3, Outer Shareable
AArch64	TLBI VMALLE1, TLBI VMALLE1NXS	TLB Invalidate by VMID, All at stage 1, EL1
AArch64	TLBI VMALLE1IS, TLBI VMALLE1ISNXS	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
AArch64	TLBI VMALLE1OS, TLBI VMALLE1OSNXS	TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable
AArch64	TLBI VMALLS12E1, TLBI VMALLS12E1NXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
AArch64	TLBI VMALLS12E1IS, TLBI VMALLS12E1ISNXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
AArch64	TLBI VMALLS12E1OS, TLBI VMALLS12E1OSNXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable
AArch64	TLBIP IPAS2E1, TLBIP IPAS2E1NXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

Exec state	Name	Description
AArch64	TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBIP RVAAE1, TLBIP RVAAE1NXS	TLB Range Invalidate by VA, All ASID, EL1
AArch64	TLBIP RVAAE1IS, TLBIP RVAAE1ISNXS	TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable
AArch64	TLBIP RVAAE1OS, TLBIP RVAAE1OSNXS	TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable
AArch64	TLBIP RVAALE1, TLBIP RVAALE1NXS	TLB Range Invalidate by VA, All ASID, Last level, EL1
AArch64	TLBIP RVAALE1IS, TLBIP RVAALE1ISNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
AArch64	TLBIP RVAALE1OS, TLBIP RVAALE1OSNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
AArch64	TLBIP RVAE1, TLBIP RVAE1NXS	TLB Range Invalidate by VA, EL1
AArch64	TLBIP RVAE1IS, TLBIP RVAE1ISNXS	TLB Range Invalidate by VA, EL1, Inner Shareable
AArch64	TLBIP RVAE1OS, TLBIP RVAE1OSNXS	TLB Range Invalidate by VA, EL1, Outer Shareable
AArch64	TLBIP RVAE2, TLBIP RVAE2NXS	TLB Range Invalidate by VA, EL2
AArch64	TLBIP RVAE2IS, TLBIP RVAE2ISNXS	TLB Range Invalidate by VA, EL2, Inner Shareable
AArch64	TLBIP RVAE2OS, TLBIP RVAE2OSNXS	TLB Range Invalidate by VA, EL2, Outer Shareable
AArch64	TLBIP RVAE3, TLBIP RVAE3NXS	TLB Range Invalidate by VA, EL3
AArch64	TLBIP RVAE3IS, TLBIP RVAE3ISNXS	TLB Range Invalidate by VA, EL3, Inner Shareable
AArch64	TLBIP RVAE3OS, TLBIP RVAE3OSNXS	TLB Range Invalidate by VA, EL3, Outer Shareable
AArch64	TLBIP RVALE1, TLBIP RVALE1NXS	TLB Range Invalidate by VA, Last level, EL1
AArch64	TLBIP RVALE1IS, TLBIP RVALE1ISNXS	TLB Range Invalidate by VA, Last level, EL1, Inner Shareable
AArch64	TLBIP RVALE1OS, TLBIP RVALE1OSNXS	TLB Range Invalidate by VA, Last level, EL1, Outer Shareable
AArch64	TLBIP RVALE2, TLBIP RVALE2NXS	TLB Range Invalidate by VA, Last level, EL2
AArch64	TLBIP RVALE2IS, TLBIP RVALE2ISNXS	TLB Range Invalidate by VA, Last level, EL2, Inner Shareable
AArch64	TLBIP RVALE2OS, TLBIP RVALE2OSNXS	TLB Range Invalidate by VA, Last level, EL2, Outer Shareable
AArch64	TLBIP RVALE3, TLBIP RVALE3NXS	TLB Range Invalidate by VA, Last level, EL3
AArch64	TLBIP RVALE3IS, TLBIP RVALE3ISNXS	TLB Range Invalidate by VA, Last level, EL3, Inner Shareable
AArch64	TLBIP RVALE3OS, TLBIP RVALE3OSNXS	TLB Range Invalidate by VA, Last level, EL3, Outer Shareable
AArch64	TLBIP VAAE1, TLBIP VAAE1NXS	TLB Invalidate Pair by VA, All ASID, EL1

Exec state	Name	Description
AArch64	TLBIP VAAE1IS, TLBIP VAAE1ISNXS	TLB Invalidate Pair by VA, All ASID, EL1, Inner Shareable
AArch64	TLBIP VAAE1OS, TLBIP VAAE1OSNXS	TLB Invalidate Pair by VA, All ASID, EL1, Outer Shareable
AArch64	TLBIP VAALE1, TLBIP VAALE1NXS	TLB Invalidate Pair by VA, All ASID, Last level, EL1
AArch64	TLBIP VAALE1IS, TLBIP VAALE1ISNXS	TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Inner Shareable
AArch64	TLBIP VAALE1OS, TLBIP VAALE1OSNXS	TLB Invalidate Pair by VA, All ASID, Last Level, EL1, Outer Shareable
AArch64	TLBIP VAE1, TLBIP VAE1NXS	TLB Invalidate Pair by VA, EL1
AArch64	TLBIP VAE1IS, TLBIP VAE1ISNXS	TLB Invalidate Pair by VA, EL1, Inner Shareable
AArch64	TLBIP VAE1OS, TLBIP VAE1OSNXS	TLB Invalidate Pair by VA, EL1, Outer Shareable
AArch64	TLBIP VAE2, TLBIP VAE2NXS	TLB Invalidate Pair by VA, EL2
AArch64	TLBIP VAE2IS, TLBIP VAE2ISNXS	TLB Invalidate Pair by VA, EL2, Inner Shareable
AArch64	TLBIP VAE2OS, TLBIP VAE2OSNXS	TLB Invalidate Pair by VA, EL2, Outer Shareable
AArch64	TLBIP VAE3, TLBIP VAE3NXS	TLB Invalidate Pair by VA, EL3
AArch64	TLBIP VAE3IS, TLBIP VAE3ISNXS	TLB Invalidate Pair by VA, EL3, Inner Shareable
AArch64	TLBIP VAE3OS, TLBIP VAE3OSNXS	TLB Invalidate Pair by VA, EL3, Outer Shareable
AArch64	TLBIP VALE1, TLBIP VALE1NXS	TLB Invalidate Pair by VA, Last level, EL1
AArch64	TLBIP VALE1IS, TLBIP VALE1ISNXS	TLB Invalidate Pair by VA, Last level, EL1, Inner Shareable
AArch64	TLBIP VALE1OS, TLBIP VALE1OSNXS	TLB Invalidate Pair by VA, Last level, EL1, Outer Shareable
AArch64	TLBIP VALE2, TLBIP VALE2NXS	TLB Invalidate Pair by VA, Last level, EL2
AArch64	TLBIP VALE2IS, TLBIP VALE2ISNXS	TLB Invalidate Pair by VA, Last level, EL2, Inner Shareable
AArch64	TLBIP VALE2OS, TLBIP VALE2OSNXS	TLB Invalidate Pair by VA, Last level, EL2, Outer Shareable
AArch64	TLBIP VALE3, TLBIP VALE3NXS	TLB Invalidate Pair by VA, Last level, EL3
AArch64	TLBIP VALE3IS, TLBIP VALE3ISNXS	TLB Invalidate Pair by VA, Last level, EL3, Inner Shareable
AArch64	TLBIP VALE3OS, TLBIP VALE3OSNXS	TLB Invalidate Pair by VA, Last level, EL3, Outer Shareable

In the PMU functional group:

Exec state	Name	Description
AArch32	PMCCFILTR	Performance Monitors Cycle Count Filter Register
AArch32	PMCCNTR	Performance Monitors Cycle Count Register
AArch32	PMCEID0	Performance Monitors Common Event Identification register 0
AArch32	PMCEID1	Performance Monitors Common Event Identification register 1
AArch32	PMCEID2	Performance Monitors Common Event Identification register 2
AArch32	PMCEID3	Performance Monitors Common Event Identification register 3
AArch32	PMCNTENCLR	Performance Monitors Count Enable Clear register
AArch32	PMCNTENSET	Performance Monitors Count Enable Set register
AArch32	PMCR	Performance Monitors Control Register
AArch32	PMEVCNTR<n>	Performance Monitors Event Count Registers
AArch32	PMEVTPER<n>	Performance Monitors Event Type Registers
AArch32	PMINTENCLR	Performance Monitors Interrupt Enable Clear register
AArch32	PMINTENSET	Performance Monitors Interrupt Enable Set register
AArch32	PMMIR	Performance Monitors Machine Identification Register
AArch32	PMOVSr	Performance Monitors Overflow Flag Status Register
AArch32	PMOVSSET	Performance Monitors Overflow Flag Status Set register
AArch32	PMSELR	Performance Monitors Event Counter Selection Register
AArch32	PMSWINC	Performance Monitors Software Increment register
AArch32	PMUSERENR	Performance Monitors User Enable Register

Exec state	Name	Description
AArch32	PMXVCNTR	Performance Monitors Selected Event Count Register
AArch32	PMXEVTYPER	Performance Monitors Selected Event Type Register
AArch64	PMCCFILTR_EL0	Performance Monitors Cycle Count Filter Register
AArch64	PMCCNTR_EL0	Performance Monitors Cycle Count Register
AArch64	PMCCNTSVR_EL1	Performance Monitors Cycle Count Saved Value Register
AArch64	PMCEID0_EL0	Performance Monitors Common Event Identification register 0
AArch64	PMCEID1_EL0	Performance Monitors Common Event Identification register 1
AArch64	PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register
AArch64	PMCNTENSET_EL0	Performance Monitors Count Enable Set register
AArch64	PMCR_EL0	Performance Monitors Control Register
AArch64	PMEVCNTR<n>_EL0	Performance Monitors Event Count Registers
AArch64	PMEVCNTSVR<n>_EL1	Performance Monitors Event Count Saved Value Register <n>
AArch64	PMEVTYPER<n>_EL0	Performance Monitors Event Type Registers
AArch64	PMICFILTR_EL0	Performance Monitors Instruction Counter Filter Register
AArch64	PMICNTR_EL0	Performance Monitors Instruction Counter Register
AArch64	PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register
AArch64	PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register
AArch64	PMMIR_EL1	Performance Monitors Machine Identification Register
AArch64	PMOVSCCLR_EL0	Performance Monitors Overflow Flag Status Clear Register
AArch64	PMOVSSSET_EL0	Performance Monitors Overflow Flag Status Set register
AArch64	PMSELR_EL0	Performance Monitors Event Counter Selection Register
AArch64	PMSWINC_EL0	Performance Monitors Software Increment register
AArch64	PMUSERENR_EL0	Performance Monitors User Enable Register
AArch64	PMXVCNTR_EL0	Performance Monitors Selected Event Count Register
AArch64	PMXEVTYPER_EL0	Performance Monitors Selected Event Type Register
External	PMAUTHSTATUS	Performance Monitors Authentication Status register
External	PMCCFILTR_EL0	Performance Monitors Cycle Counter Filter Register
External	PMCCIDSR	CONTEXTIDR ELx Sample Register
External	PMCCNTR_EL0	Performance Monitors Cycle Counter
External	PMCCNTSVR_EL1	Performance Monitors Cycle Count Saved Value Register
External	PMCEID0	Performance Monitors Common Event Identification register 0
External	PMCEID1	Performance Monitors Common Event Identification register 1
External	PMCEID2	Performance Monitors Common Event Identification register 2
External	PMCEID3	Performance Monitors Common Event Identification register 3
External	PMCFGR	Performance Monitors Configuration Register
External	PMCGCR0	Counter Group Configuration Register 0
External	PMCID1SR	CONTEXTIDR EL1 Sample Register
External	PMCID2SR	CONTEXTIDR EL2 Sample Register
External	PMCIDR0	Performance Monitors Component Identification Register 0
External	PMCIDR1	Performance Monitors Component Identification Register 1
External	PMCIDR2	Performance Monitors Component Identification Register 2
External	PMCIDR3	Performance Monitors Component Identification Register 3
External	PMCNTEN	Performance Monitors Count Enable register
External	PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register
External	PMCNTENSET_EL0	Performance Monitors Count Enable Set register
External	PMCR_EL0	Performance Monitors Control Register
External	PMDEVAFF	Performance Monitors Device Affinity register
External	PMDEVAFF0	Performance Monitors Device Affinity register 0
External	PMDEVAFF1	Performance Monitors Device Affinity register 1
External	PMDEVARCH	Performance Monitors Device Architecture register
External	PMDEVID	Performance Monitors Device ID register
External	PMDEVTYPE	Performance Monitors Device Type register
External	PMEVCNTR<n>_EL0	Performance Monitors Event Count Registers
External	PMEVCNTSVR<n>_EL1	Performance Monitors Event Count Saved Value Register <n>
External	PMEVTYPER<n>_EL0	Performance Monitors Event Type Registers
External	PMICFILTR_EL0	Performance Monitors Instruction Counter Filter Register
External	PMICNTR_EL0	Performance Monitors Instruction Counter Register
External	PMICNTSVR_EL1	Performance Monitors Instruction Count Saved Value Register
External	PMIIDR	Performance Monitors Implementation Identification Register
External	PMINTEN	Performance Monitors Interrupt Enable register
External	PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register
External	PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register
External	PMITCTRL	Performance Monitors Integration mode Control register
External	PMLAR	Performance Monitors Lock Access Register

Exec state	Name	Description
External	PMLSR	Performance Monitors Lock Status Register
External	PMMIR	Performance Monitors Machine Identification Register
External	PMOVS	Performance Monitors Overflow Flag Status register
External	PMOVSCLR_EL0	Performance Monitors Overflow Flag Status Clear register
External	PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set register
External	PMPCCTL	PC Sample-based Profiling Control Register
External	PMPCSR	Program Counter Sample Register
External	PMPIDR0	Performance Monitors Peripheral Identification Register 0
External	PMPIDR1	Performance Monitors Peripheral Identification Register 1
External	PMPIDR2	Performance Monitors Peripheral Identification Register 2
External	PMPIDR3	Performance Monitors Peripheral Identification Register 3
External	PMPIDR4	Performance Monitors Peripheral Identification Register 4
External	PMSSCR_EL1	Performance Monitors Snapshot Status and Capture Register
External	PMSWINC_EL0	Performance Monitors Software Increment register
External	PMVCIISR	CONTEXTIDR_EL1 and VMID Sample Register
External	PMVIDSR	VMID Sample Register

In the Reset functional group:

Exec state	Name	Description
AArch32	HRMR	Hyp Reset Management Register
AArch32	RMR	Reset Management Register
AArch32	RVBAR	Reset Vector Base Address Register
AArch64	RMR_EL1	Reset Management Register (EL1)
AArch64	RMR_EL2	Reset Management Register (EL2)
AArch64	RMR_EL3	Reset Management Register (EL3)
AArch64	RVBAR_EL1	Reset Vector Base Address Register (if EL2 and EL3 not implemented)
AArch64	RVBAR_EL2	Reset Vector Base Address Register (if EL3 not implemented)
AArch64	RVBAR_EL3	Reset Vector Base Address Register (if EL3 implemented)

In the Thread functional group:

Exec state	Name	Description
AArch32	HTPIDR	Hyp Software Thread ID Register
AArch32	TPIDRPRW	PL1 Software Thread ID Register
AArch32	TPIDRURO	PL0 Read-Only Software Thread ID Register
AArch32	TPIDRURW	PL0 Read/Write Software Thread ID Register
AArch64	SCXTNUM_EL0	EL0 Read/Write Software Context Number
AArch64	SCXTNUM_EL1	EL1 Read/Write Software Context Number
AArch64	SCXTNUM_EL2	EL2 Read/Write Software Context Number
AArch64	SCXTNUM_EL3	EL3 Read/Write Software Context Number
AArch64	TPIDR2_EL0	EL0 Read/Write Software Thread ID Register 2
AArch64	TPIDRRO_EL0	EL0 Read-Only Software Thread ID Register
AArch64	TPIDR_EL0	EL0 Read/Write Software Thread ID Register
AArch64	TPIDR_EL1	EL1 Software Thread ID Register
AArch64	TPIDR_EL2	EL2 Software Thread ID Register
AArch64	TPIDR_EL3	EL3 Software Thread ID Register

In the IMP DEF functional group:

Exec state	Name	Description
AArch32	ACTLR	Auxiliary Control Register
AArch32	ACTLR2	Auxiliary Control Register 2
AArch32	ADFSR	Auxiliary Data Fault Status Register
AArch32	AIDR	Auxiliary ID Register
AArch32	AIFSR	Auxiliary Instruction Fault Status Register
AArch32	AMAIRO	Auxiliary Memory Attribute Indirection Register 0

Exec state	Name	Description
AArch32	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
AArch32	HACTLR	Hyp Auxiliary Control Register
AArch32	HACTLR2	Hyp Auxiliary Control Register 2
AArch32	HADFSR	Hyp Auxiliary Data Fault Status Register
AArch32	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
AArch32	HAMAIRO	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch64	ACTLR_EL1	Auxiliary Control Register (EL1)
AArch64	ACTLR_EL2	Auxiliary Control Register (EL2)
AArch64	ACTLR_EL3	Auxiliary Control Register (EL3)
AArch64	AFSR0_EL1	Auxiliary Fault Status Register 0 (EL1)
AArch64	AFSR0_EL2	Auxiliary Fault Status Register 0 (EL2)
AArch64	AFSR0_EL3	Auxiliary Fault Status Register 0 (EL3)
AArch64	AFSR1_EL1	Auxiliary Fault Status Register 1 (EL1)
AArch64	AFSR1_EL2	Auxiliary Fault Status Register 1 (EL2)
AArch64	AFSR1_EL3	Auxiliary Fault Status Register 1 (EL3)
AArch64	AIDR_EL1	Auxiliary ID Register
AArch64	AMAIR_EL1	Auxiliary Memory Attribute Indirection Register (EL1)
AArch64	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	AMAIR_EL3	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	HACR_EL2	Hypervisor Auxiliary Control Register
AArch64	S3_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED registers
AArch64	SYS S1_<op1>_<Cn>_<Cm>_<op2> , SYS S1_<op1>_<Cn>_<Cm>_<op2> , SYSP S1_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED maintenance instructions

In the Timer functional group:

Exec state	Name	Description
AArch32	CNTFRQ	Counter-timer Frequency register
AArch32	CNTHPS_CTL	Counter-timer Secure Physical Timer Control Register (EL2)
AArch32	CNTHPS_CVAL	Counter-timer Secure Physical Timer CompareValue Register (EL2)
AArch32	CNTHPS_TVAL	Counter-timer Secure Physical Timer TimerValue Register (EL2)
AArch32	CNTHP_CTL	Counter-timer Hyp Physical Timer Control register
AArch32	CNTHVS_CTL	Counter-timer Secure Virtual Timer Control Register (EL2)
AArch32	CNTHVS_CVAL	Counter-timer Secure Virtual Timer CompareValue Register (EL2)
AArch32	CNTHVS_TVAL	Counter-timer Secure Virtual Timer TimerValue Register (EL2)
AArch32	CNTHV_CTL	Counter-timer Virtual Timer Control register (EL2)
AArch32	CNTHV_CVAL	Counter-timer Virtual Timer CompareValue register (EL2)
AArch32	CNTHV_TVAL	Counter-timer Virtual Timer TimerValue register (EL2)
AArch32	CNTKCTL	Counter-timer Kernel Control register
AArch32	CNTPCT	Counter-timer Physical Count register
AArch32	CNTPCTSS	Counter-timer Self-Synchronized Physical Count register
AArch32	CNTP_CTL	Counter-timer Physical Timer Control register

Exec state	Name	Description
AArch32	CNTP_CVAL	Counter-timer Physical Timer CompareValue register
AArch32	CNTP_TVAL	Counter-timer Physical Timer TimerValue register
AArch32	CNTVCT	Counter-timer Virtual Count register
AArch32	CNTVCTSS	Counter-timer Self-Synchronized Virtual Count register
AArch32	CNTV_CTL	Counter-timer Virtual Timer Control register
AArch32	CNTV_CVAL	Counter-timer Virtual Timer CompareValue register
AArch32	CNTV_TVAL	Counter-timer Virtual Timer TimerValue register
AArch64	CNTFRQ_EL0	Counter-timer Frequency register
AArch64	CNTHVS_CTL_EL2	Counter-timer Secure Virtual Timer Control register (EL2)
AArch64	CNTHVS_CVAL_EL2	Counter-timer Secure Virtual Timer CompareValue register (EL2)
AArch64	CNTHVS_TVAL_EL2	Counter-timer Secure Virtual Timer TimerValue register (EL2)
AArch64	CNTHV_CTL_EL2	Counter-timer Virtual Timer Control register (EL2)
AArch64	CNTHV_CVAL_EL2	Counter-timer Virtual Timer CompareValue register (EL2)
AArch64	CNTHV_TVAL_EL2	Counter-timer Virtual Timer TimerValue Register (EL2)
AArch64	CNTKCTL_EL1	Counter-timer Kernel Control register
AArch64	CNTPCTSS_EL0	Counter-timer Self-Synchronized Physical Count register
AArch64	CNTPCT_EL0	Counter-timer Physical Count register
AArch64	CNTPOFF_EL2	Counter-timer Physical Offset register
AArch64	CNTPS_CTL_EL1	Counter-timer Physical Secure Timer Control register
AArch64	CNTPS_CVAL_EL1	Counter-timer Physical Secure Timer CompareValue register
AArch64	CNTPS_TVAL_EL1	Counter-timer Physical Secure Timer TimerValue register
AArch64	CNTP_CTL_EL0	Counter-timer Physical Timer Control register
AArch64	CNTP_CVAL_EL0	Counter-timer Physical Timer CompareValue register
AArch64	CNTP_TVAL_EL0	Counter-timer Physical Timer TimerValue register
AArch64	CNTVCTSS_EL0	Counter-timer Self-Synchronized Virtual Count register
AArch64	CNTVCT_EL0	Counter-timer Virtual Count register
AArch64	CNTV_CTL_EL0	Counter-timer Virtual Timer Control register
AArch64	CNTV_CVAL_EL0	Counter-timer Virtual Timer CompareValue register
AArch64	CNTV_TVAL_EL0	Counter-timer Virtual Timer TimerValue register
External	CNTACR<n>	Counter-timer Access Control Registers
External	CNTCR	Counter Control Register
External	CNTCV	Counter Count Value register
External	CNTEL0ACR	Counter-timer EL0 Access Control Register
External	CNTFID0	Counter Frequency ID
External	CNTFID<n>	Counter Frequency IDs, n > 0
External	CNTFRQ	Counter-timer Frequency
External	CNTID	Counter Identification Register
External	CNTNSAR	Counter-timer Non-secure Access Register
External	CNTPCT	Counter-timer Physical Count
External	CNTP_CTL	Counter-timer Physical Timer Control
External	CNTP_CVAL	Counter-timer Physical Timer CompareValue
External	CNTP_TVAL	Counter-timer Physical Timer TimerValue
External	CNTSCR	Counter Scale Register
External	CNTSR	Counter Status Register
External	CNTTIDR	Counter-timer Timer ID Register
External	CNTVCT	Counter-timer Virtual Count
External	CNTVOFF	Counter-timer Virtual Offset
External	CNTVOFF<n>	Counter-timer Virtual Offsets
External	CNTV_CTL	Counter-timer Virtual Timer Control
External	CNTV_CVAL	Counter-timer Virtual Timer CompareValue
External	CNTV_TVAL	Counter-timer Virtual Timer TimerValue
External	CounterID<n>	Counter ID registers

In the Debug functional group:

Exec state	Name	Description
AArch32	DBGAUTHSTATUS	Debug Authentication Status register
AArch32	DBGBCR<n>	Debug Breakpoint Control Registers
AArch32	DBGBVR<n>	Debug Breakpoint Value Registers
AArch32	DBGBXVR<n>	Debug Breakpoint Extended Value Registers
AArch32	DBGCLAIMCLR	Debug CLAIM Tag Clear register
AArch32	DBGCLAIMSET	Debug CLAIM Tag Set register
AArch32	DBGDCCINT	DCC Interrupt Enable Register

Exec state	Name	Description
AArch32	DBGDEVID	Debug Device ID register 0
AArch32	DBGDEVID1	Debug Device ID register 1
AArch32	DBGDEVID2	Debug Device ID register 2
AArch32	DBGDIDR	Debug ID Register
AArch32	DBGDRAR	Debug ROM Address Register
AArch32	DBGDSAR	Debug Self Address Register
AArch32	DBGDSCRext	Debug Status and Control Register, External View
AArch32	DBGDSCRint	Debug Status and Control Register, Internal View
AArch32	DBGDTRRXext	Debug OS Lock Data Transfer Register, Receive, External View
AArch32	DBGDTRRXint	Debug Data Transfer Register, Receive
AArch32	DBGDTRTXext	Debug OS Lock Data Transfer Register, Transmit
AArch32	DBGDTRTXint	Debug Data Transfer Register, Transmit
AArch32	DBGOSDLR	Debug OS Double Lock Register
AArch32	DBGOSECCR	Debug OS Lock Exception Catch Control Register
AArch32	DBGOSLAR	Debug OS Lock Access Register
AArch32	DBGOSLSR	Debug OS Lock Status Register
AArch32	DBGPRCR	Debug Power Control Register
AArch32	DBGVCR	Debug Vector Catch Register
AArch32	DBGWCR<n>	Debug Watchpoint Control Registers
AArch32	DBGWFAR	Debug Watchpoint Fault Address Register
AArch32	DBGWVR<n>	Debug Watchpoint Value Registers
AArch32	TRFCR	Trace Filter Control Register
AArch64	DBGAUTHSTATUS_EL1	Debug Authentication Status register
AArch64	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
AArch64	DBGBVR<n>_EL1	Debug Breakpoint Value Registers
AArch64	DBGCLAIMCLR_EL1	Debug CLAIM Tag Clear register
AArch64	DBGCLAIMSET_EL1	Debug CLAIM Tag Set register
AArch64	DBGDTRRX_EL0	Debug Data Transfer Register, Receive
AArch64	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit
AArch64	DBGDTR_EL0	Debug Data Transfer Register, half-duplex
AArch64	DBGPRCR_EL1	Debug Power Control Register
AArch64	DBGVCR32_EL2	Debug Vector Catch Register
AArch64	DBGWCR<n>_EL1	Debug Watchpoint Control Registers
AArch64	DBGWVR<n>_EL1	Debug Watchpoint Value Registers
AArch64	DLR_EL0	Debug Link Register
AArch64	DSPSR_EL0	Debug Saved Program Status Register
AArch64	MDCCINT_EL1	Monitor DCC Interrupt Enable Register
AArch64	MDCCSR_EL0	Monitor DCC Status Register
AArch64	MDRAR_EL1	Monitor Debug ROM Address Register
AArch64	MDSCR_EL1	Monitor Debug System Control Register
AArch64	OSDLR_EL1	OS Double Lock Register
AArch64	OSDTRRX_EL1	OS Lock Data Transfer Register, Receive
AArch64	OSDTRTX_EL1	OS Lock Data Transfer Register, Transmit
AArch64	OSECCR_EL1	OS Lock Exception Catch Control Register
AArch64	OSLAR_EL1	OS Lock Access Register
AArch64	OSLSR_EL1	OS Lock Status Register
AArch64	TRFCR_EL1	Trace Filter Control Register (EL1)
AArch64	TRFCR_EL2	Trace Filter Control Register (EL2)
External	DBGAUTHSTATUS_EL1	Debug Authentication Status register
External	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
External	DBGBVR<n>_EL1	Debug Breakpoint Value Registers
External	DBGCLAIMCLR_EL1	Debug CLAIM Tag Clear register
External	DBGCLAIMSET_EL1	Debug CLAIM Tag Set register
External	DBGDTRRX_EL0	Debug Data Transfer Register, Receive
External	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit
External	DBGWCR<n>_EL1	Debug Watchpoint Control Registers
External	DBGWVR<n>_EL1	Debug Watchpoint Value Registers
External	EDACR	External Debug Auxiliary Control Register
External	EDCIDR0	External Debug Component Identification Register 0
External	EDCIDR1	External Debug Component Identification Register 1
External	EDCIDR2	External Debug Component Identification Register 2
External	EDCIDR3	External Debug Component Identification Register 3
External	EDCIDS	External Debug Context ID Sample Register
External	EDDEVAFF0	External Debug Device Affinity register 0

Exec state	Name	Description
External	EDDEVAFF1	External Debug Device Affinity register 1
External	EDDEVARCH	External Debug Device Architecture register
External	EDDEVID	External Debug Device ID register 0
External	EDDEVID1	External Debug Device ID register 1
External	EDDEVID2	External Debug Device ID register 2
External	EDDEVTYPE	External Debug Device Type register
External	EDDFR1	External Debug Feature Register 1
External	EDECCR	External Debug Exception Catch Control Register
External	EDECR	External Debug Execution Control Register
External	EDESR	External Debug Event Status Register
External	EDHSR	External Debug HaltingHalt SyndromeStatus Register
External	EDITCTRL	External Debug Integration mode Control register
External	EDITR	External Debug Instruction Transfer Register
External	EDLAR	External Debug Lock Access Register
External	EDLSR	External Debug Lock Status Register
External	EDPCSR	External Debug Program Counter Sample Register
External	EDPIDR0	External Debug Peripheral Identification Register 0
External	EDPIDR1	External Debug Peripheral Identification Register 1
External	EDPIDR2	External Debug Peripheral Identification Register 2
External	EDPIDR3	External Debug Peripheral Identification Register 3
External	EDPIDR4	External Debug Peripheral Identification Register 4
External	EDPRCR	External Debug Power/Reset Control Register
External	EDPRSR	External Debug Processor Status Register
External	EDRCR	External Debug Reserve Control Register
External	EDSCR	External Debug Status and Control Register
External	EDSCR2	External Debug Status and Control Register 2
External	EDVIDSR	External Debug Virtual Context Sample Register
External	EDWAR	External Debug Watchpoint Address Register
External	OSLAR_EL1	OS Lock Access Register

In the CTI functional group:

Exec state	Name	Description
External	ASICCTL	CTI External Multiplexer Control register
External	CTIAPPCLEAR	CTI Application Trigger Clear register
External	CTIAPPULSE	CTI Application Pulse register
External	CTIAPPSET	CTI Application Trigger Set register
External	CTIAUTHSTATUS	CTI Authentication Status register
External	CTICHINSTATUS	CTI Channel In Status register
External	CTICHOUTSTATUS	CTI Channel Out Status register
External	CTICIDR0	CTI Component Identification Register 0
External	CTICIDR1	CTI Component Identification Register 1
External	CTICIDR2	CTI Component Identification Register 2
External	CTICIDR3	CTI Component Identification Register 3
External	CTICLAIMCLR	CTI CLAIM Tag Clear register
External	CTICLAIMSET	CTI CLAIM Tag Set register
External	CTICONTROL	CTI Control register
External	CTIDEVAFF0	CTI Device Affinity register 0
External	CTIDEVAFF1	CTI Device Affinity register 1
External	CTIDEVARCH	CTI Device Architecture register
External	CTIDEVCTL	CTI Device Control register
External	CTIDEVID	CTI Device ID register 0
External	CTIDEVID1	CTI Device ID register 1
External	CTIDEVID2	CTI Device ID register 2
External	CTIDEVTYPE	CTI Device Type register
External	CTIGATE	CTI Channel Gate Enable register
External	CTIINEN<n>	CTI Input Trigger to Output Channel Enable registers
External	CTIINTACK	CTI Output Trigger Acknowledge register
External	CTIITCTRL	CTI Integration mode Control register
External	CTILAR	CTI Lock Access Register
External	CTILSR	CTI Lock Status Register
External	CTIOUTEN<n>	CTI Input Channel to Output Trigger Enable registers
External	CTIPIDR0	CTI Peripheral Identification Register 0

Exec state	Name	Description
External	CTIPIDR1	CTI Peripheral Identification Register 1
External	CTIPIDR2	CTI Peripheral Identification Register 2
External	CTIPIDR3	CTI Peripheral Identification Register 3
External	CTIPIDR4	CTI Peripheral Identification Register 4
External	CTITRIGINSTATUS	CTI Trigger In Status register
External	CTITRIGOUTSTATUS	CTI Trigger Out Status register

In the Virt functional group:

Exec state	Name	Description
AArch32	ATS1HR	Address Translate Stage 1 Hyp mode Read
AArch32	ATS1HW	Address Translate Stage 1 Hyp mode Write
AArch32	CNTHCTL	Counter-timer Hyp Control register
AArch32	CNTHP_CVAL	Counter-timer Hyp Physical CompareValue register
AArch32	CNTHP_TVAL	Counter-timer Hyp Physical Timer TimerValue register
AArch32	CNTVOFF	Counter-timer Virtual Offset register
AArch32	HACR	Hyp Auxiliary Configuration Register
AArch32	HACTLR	Hyp Auxiliary Control Register
AArch32	HACTLR2	Hyp Auxiliary Control Register 2
AArch32	HADFSR	Hyp Auxiliary Data Fault Status Register
AArch32	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
AArch32	HAMAIRO	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch32	HCPTR	Hyp Architectural Feature Trap Register
AArch32	HCR	Hyp Configuration Register
AArch32	HCR2	Hyp Configuration Register 2
AArch32	HDCR	Hyp Debug Control Register
AArch32	HDFAR	Hyp Data Fault Address Register
AArch32	HIFAR	Hyp Instruction Fault Address Register
AArch32	HMAIRO	Hyp Memory Attribute Indirection Register 0
AArch32	HMAIR1	Hyp Memory Attribute Indirection Register 1
AArch32	HPFAR	Hyp IPA Fault Address Register
AArch32	HRMR	Hyp Reset Management Register
AArch32	HSCTLR	Hyp System Control Register
AArch32	HSR	Hyp Syndrome Register
AArch32	HSTR	Hyp System Trap Register
AArch32	HTCR	Hyp Translation Control Register
AArch32	HTPIDR	Hyp Software Thread ID Register
AArch32	HTRFCR	Hyp Trace Filter Control Register
AArch32	HTTBR	Hyp Translation Table Base Register
AArch32	HVBAR	Hyp Vector Base Address Register
AArch32	ICC_HSRE	Interrupt Controller Hyp System Register Enable register
AArch32	ICH_AP0R<n>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch32	ICH_AP1R<n>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch32	ICH_EISR	Interrupt Controller End of Interrupt Status Register
AArch32	ICH_ELRSR	Interrupt Controller Empty List Register Status Register
AArch32	ICH_HCR	Interrupt Controller Hyp Control Register
AArch32	ICH_LR<n>	Interrupt Controller List Registers
AArch32	ICH_LRC<n>	Interrupt Controller List Registers
AArch32	ICH_MISR	Interrupt Controller Maintenance Interrupt State Register
AArch32	ICH_VMCR	Interrupt Controller Virtual Machine Control Register
AArch32	ICH_VTR	Interrupt Controller VGIC Type Register
AArch32	TLBIALH	TLB Invalidate All, Hyp mode
AArch32	TLBIALHIS	TLB Invalidate All, Hyp mode, Inner Shareable
AArch32	TLBIIPAS2	TLB Invalidate by Intermediate Physical Address, Stage 2
AArch32	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
AArch32	TLBIIPAS2L	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
AArch32	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
AArch32	TLBIMVAH	TLB Invalidate by VA, Hyp mode
AArch32	TLBIMVAHIS	TLB Invalidate by VA, Hyp mode, Inner Shareable

Exec state	Name	Description
AArch32	TLBIMVALH	TLB Invalidate by VA, Last level, Hyp mode
AArch32	TLBIMVALHIS	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
AArch32	VMPIDR	Virtualization Multiprocessor ID Register
AArch32	VPIDR	Virtualization Processor ID Register
AArch32	VTCT	Virtualization Translation Control Register
AArch32	VTTBR	Virtualization Translation Table Base Register
AArch64	ACTLR_EL2	Auxiliary Control Register (EL2)
AArch64	AFSR0_EL2	Auxiliary Fault Status Register 0 (EL2)
AArch64	AFSR1_EL2	Auxiliary Fault Status Register 1 (EL2)
AArch64	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	CNTHTCTL_EL2	Counter-timer Hypervisor Control register
AArch64	CNTHPS_CTL_EL2	Counter-timer Secure Physical Timer Control register (EL2)
AArch64	CNTHPS_CVAL_EL2	Counter-timer Secure Physical Timer CompareValue register (EL2)
AArch64	CNTHPS_TVAL_EL2	Counter-timer Secure Physical Timer TimerValue register (EL2)
AArch64	CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register
AArch64	CNTHP_CVAL_EL2	Counter-timer Physical Timer CompareValue register (EL2)
AArch64	CNTHP_TVAL_EL2	Counter-timer Physical Timer TimerValue register (EL2)
AArch64	CNTVOFF_EL2	Counter-timer Virtual Offset register
AArch64	CPTR_EL2	Architectural Feature Trap Register (EL2)
AArch64	ESR_EL2	Exception Syndrome Register (EL2)
AArch64	FAR_EL2	Fault Address Register (EL2)
AArch64	HACR_EL2	Hypervisor Auxiliary Control Register
AArch64	HCRX_EL2	Extended Hypervisor Configuration Register
AArch64	HCR_EL2	Hypervisor Configuration Register
AArch64	HFGITR2_EL2	Hypervisor Fine-Grained Instruction Trap Register 2
AArch64	HPFAR_EL2	Hypervisor IPA Fault Address Register
AArch64	HSTR_EL2	Hypervisor System Trap Register
AArch64	ICC_SRE_EL2	Interrupt Controller System Register Enable register (EL2)
AArch64	ICH_AP0R<n>_EL2	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch64	ICH_AP1R<n>_EL2	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch64	ICH_EISR_EL2	Interrupt Controller End of Interrupt Status Register
AArch64	ICH_ELRSR_EL2	Interrupt Controller Empty List Register Status Register
AArch64	ICH_HCR_EL2	Interrupt Controller Hyp Control Register
AArch64	ICH_LR<n>_EL2	Interrupt Controller List Registers
AArch64	ICH_MISR_EL2	Interrupt Controller Maintenance Interrupt State Register
AArch64	ICH_VMCR_EL2	Interrupt Controller Virtual Machine Control Register
AArch64	ICH_VTR_EL2	Interrupt Controller VGIC Type Register
AArch64	MAIR_EL2	Memory Attribute Indirection Register (EL2)
AArch64	MDCR_EL2	Monitor Debug Configuration Register (EL2)
AArch64	RMR_EL2	Reset Management Register (EL2)
AArch64	SCTLR2_EL2	System Control Register (EL2)
AArch64	SCTLR_EL2	System Control Register (EL2)
AArch64	TCR2_EL2	Extended Translation Control Register (EL2)
AArch64	TCR_EL2	Translation Control Register (EL2)
AArch64	TLBI_IPAS2E1, TLBI_IPAS2E1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBI_IPAS2E1IS, TLBI_IPAS2E1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBI_IPAS2E1IOS, TLBI_IPAS2E1IOSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBI_IPAS2LE1, TLBI_IPAS2LE1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBI_IPAS2LE1IS, TLBI_IPAS2LE1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBI_IPAS2LE1IOS, TLBI_IPAS2LE1IOSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBI_RIPAS2E1, TLBI_RIPAS2E1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBI_RIPAS2E1IS, TLBI_RIPAS2E1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBI_RIPAS2E1IOS, TLBI_RIPAS2E1IOSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBI_RIPAS2LE1, TLBI_RIPAS2LE1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

Exec state	Name	Description
AArch64	TLBI RIPAS2LE1IS, TLBI RIPAS2LE1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBI RIPAS2LE1OS, TLBI RIPAS2LE1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBIP IPAS2E1, TLBIP IPAS2E1NXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBIP IPAS2E1IS, TLBIP IPAS2E1ISNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBIP IPAS2E1OS, TLBIP IPAS2E1OSNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBIP IPAS2LE1, TLBIP IPAS2LE1NXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBIP IPAS2LE1IS, TLBIP IPAS2LE1ISNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS	TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TLBIP RIPAS2E1, TLBIP RIPAS2E1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	TLBIP RIPAS2E1IS, TLBIP RIPAS2E1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBIP RIPAS2E1OS, TLBIP RIPAS2E1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
AArch64	TLBIP RIPAS2LE1, TLBIP RIPAS2LE1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBIP RIPAS2LE1IS, TLBIP RIPAS2LE1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBIP RIPAS2LE1OS, TLBIP RIPAS2LE1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
AArch64	TPIDR_EL2	EL2 Software Thread ID Register
AArch64	TTBR0_EL2	Translation Table Base Register 0 (EL2)
AArch64	TTBR1_EL2	Translation Table Base Register 1 (EL2)
AArch64	VBAR_EL2	Vector Base Address Register (EL2)
AArch64	VMPIDR_EL2	Virtualization Multiprocessor ID Register
AArch64	VPIDR_EL2	Virtualization Processor ID Register
AArch64	VTCR_EL2	Virtualization Translation Control Register
AArch64	VTTBR_EL2	Virtualization Translation Table Base Register

In the Secure functional group:

Exec state	Name	Description
AArch32	ICC_MCTLR	Interrupt Controller Monitor Control Register
AArch32	ICC_MSRE	Interrupt Controller Monitor System Register Enable register
AArch32	MVBAR	Monitor Vector Base Address Register
AArch32	NSACR	Non-Secure Access Control Register
AArch32	SCR	Secure Configuration Register
AArch32	SDCR	Secure Debug Control Register
AArch32	SDER	Secure Debug Enable Register
AArch64	ACTLR_EL3	Auxiliary Control Register (EL3)
AArch64	AFSR0_EL3	Auxiliary Fault Status Register 0 (EL3)
AArch64	AFSR1_EL3	Auxiliary Fault Status Register 1 (EL3)
AArch64	AMAIR_EL3	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	CPTR_EL3	Architectural Feature Trap Register (EL3)
AArch64	ICC_CTLR_EL3	Interrupt Controller Control Register (EL3)
AArch64	ICC_SRE_EL3	Interrupt Controller System Register Enable register (EL3)
AArch64	MDCR_EL3	Monitor Debug Configuration Register (EL3)
AArch64	SCR_EL3	Secure Configuration Register
AArch64	SDER32_EL3	AArch32 Secure Debug Enable Register
AArch64	VBAR_EL3	Vector Base Address Register (EL3)

In the Float functional group:

Exec state	Name	Description
AArch32	FPEXC	Floating-Point Exception Control register
AArch32	FPSCR	Floating-Point Status and Control Register
AArch32	FPSID	Floating-Point System ID register
AArch32	MVFR0	Media and VFP Feature Register 0
AArch32	MVFR1	Media and VFP Feature Register 1
AArch32	MVFR2	Media and VFP Feature Register 2
AArch64	FPCR	Floating-point Control Register
AArch64	FPEXC32_EL2	Floating-Point Exception Control register
AArch64	FPSCR	Floating-point Status Register
AArch64	MVFR0_EL1	AArch32 Media and VFP Feature Register 0
AArch64	MVFR1_EL1	AArch32 Media and VFP Feature Register 1
AArch64	MVFR2_EL1	AArch32 Media and VFP Feature Register 2

In the Legacy functional group:

Exec state	Name	Description
AArch32	CP15DMB	Data Memory Barrier System instruction
AArch32	CP15DSB	Data Synchronization Barrier System instruction
AArch32	CP15ISB	Instruction Synchronization Barrier System instruction
AArch32	FCSEIDR	FCSE Process ID register
AArch32	JIDR	Jazelle ID Register
AArch32	JMCR	Jazelle Main Configuration Register
AArch32	JOSCR	Jazelle OS Control Register

In the Trace functional group:

Exec state	Name	Description
AArch64	TRCACATR<n>	Address Comparator Access Type Register <n>
AArch64	TRCACVR<n>	Address Comparator Value Register <n>
AArch64	TRCAUXCTLR	Auxiliary Control Register
AArch64	TRCBBCTLR	Branch Broadcast Control Register
AArch64	TRCCCCTLR	Cycle Count Control Register
AArch64	TRCCIDCCTLR0	Context Identifier Comparator Control Register 0
AArch64	TRCCIDCCTLR1	Context Identifier Comparator Control Register 1
AArch64	TRCCIDCVR<n>	Context Identifier Comparator Value Registers <n>
AArch64	TRCCLAIMCLR	Claim Tag Clear Register
AArch64	TRCCLAIMSET	Claim Tag Set Register
AArch64	TRCCNTCTLR<n>	Counter Control Register <n>
AArch64	TRCCNTRLDVR<n>	Counter Reload Value Register <n>
AArch64	TRCCNTVR<n>	Counter Value Register <n>
AArch64	TRCCONFIGR	Trace Configuration Register
AArch64	TRCEVENTCTL0R	Event Control 0 Register
AArch64	TRCEVENTCTL1R	Event Control 1 Register
AArch64	TRCEXTINSELR<n>	External Input Select Register <n>
AArch64	TRCIDR0	ID Register 0
AArch64	TRCIDR1	ID Register 1
AArch64	TRCIDR10	ID Register 10
AArch64	TRCIDR11	ID Register 11
AArch64	TRCIDR12	ID Register 12
AArch64	TRCIDR13	ID Register 13
AArch64	TRCIDR2	ID Register 2
AArch64	TRCIDR3	ID Register 3
AArch64	TRCIDR4	ID Register 4
AArch64	TRCIDR5	ID Register 5
AArch64	TRCIDR6	ID Register 6
AArch64	TRCIDR7	ID Register 7
AArch64	TRCIDR8	ID Register 8
AArch64	TRCIDR9	ID Register 9
AArch64	TRCIMSPEC0	IMP DEF Register 0

Exec state	Name	Description
AArch64	TRCIMSPEC<n>	IMP DEF Register <n>
AArch64	TRCITECR_EL1	Instrumentation Trace Control Register (EL1)
AArch64	TRCITECR_EL2	Instrumentation Trace Control Register (EL2)
AArch64	TRCITEEDCR	Instrumentation Trace Extension External Debug Control Register
AArch64	TRCPRGCTLR	Programming Control Register
AArch64	TRCQCTLR	Q Element Control Register
AArch64	TRCRSCTLR<n>	Resource Selection Control Register <n>
AArch64	TRCRSR	Resources Status Register
AArch64	TRCSEQEVR<n>	Sequencer State Transition Control Register <n>
AArch64	TRCSEQRSTEV	Sequencer Reset Control Register
AArch64	TRCSEQSTR	Sequencer State Register
AArch64	TRCSSCCR<n>	Single-shot Comparator Control Register <n>
AArch64	TRCSSCSR<n>	Single-shot Comparator Control Status Register <n>
AArch64	TRCSSPCICR<n>	Single-shot Processing Element Comparator Input Control Register <n>
AArch64	TRCSTALLCTL	Stall Control Register
AArch64	TRCSTATR	Trace Status Register
AArch64	TRCSYNCP	Synchronization Period Register
AArch64	TRCTRACEIDR	Trace ID Register
AArch64	TRCTSCTLR	Timestamp Control Register
AArch64	TRCVICTLR	ViewInst Main Control Register
AArch64	TRCVIIECTLR	ViewInst Include/Exclude Control Register
AArch64	TRCVIPCSSCTLR	ViewInst Start/Stop PE Comparator Control Register
AArch64	TRCVISSCTLR	ViewInst Start/Stop Control Register
AArch64	TRCVMIDCCTLR0	Virtual Context Identifier Comparator Control Register 0
AArch64	TRCVMIDCCTLR1	Virtual Context Identifier Comparator Control Register 1
AArch64	TRCVMIDCVR<n>	Virtual Context Identifier Comparator Value Register <n>
External	TRCACATR<n>	Address Comparator Access Type Register <n>
External	TRCACVR<n>	Address Comparator Value Register <n>
External	TRCAUXCTL	Auxiliary Control Register
External	TRCBBCTL	Branch Broadcast Control Register
External	TRCCCCTL	Cycle Count Control Register
External	TRCCIDCCTLR0	Context Identifier Comparator Control Register 0
External	TRCCIDCCTLR1	Context Identifier Comparator Control Register 1
External	TRCCIDCVR<n>	Context Identifier Comparator Value Registers <n>
External	TRCCLAIMCLR	Claim Tag Clear Register
External	TRCCLAIMSET	Claim Tag Set Register
External	TRCCNTCTL<n>	Counter Control Register <n>
External	TRCCNTRLDVR<n>	Counter Reload Value Register <n>
External	TRCCNTVR<n>	Counter Value Register <n>
External	TRCCONFIGR	Trace Configuration Register
External	TRCEVENTCTL0R	Event Control 0 Register
External	TRCEVENTCTL1R	Event Control 1 Register
External	TRCEXTINSEL<n>	External Input Select Register <n>
External	TRCIDR0	ID Register 0
External	TRCIDR1	ID Register 1
External	TRCIDR10	ID Register 10
External	TRCIDR11	ID Register 11
External	TRCIDR12	ID Register 12
External	TRCIDR13	ID Register 13
External	TRCIDR2	ID Register 2
External	TRCIDR3	ID Register 3
External	TRCIDR4	ID Register 4
External	TRCIDR5	ID Register 5
External	TRCIDR6	ID Register 6
External	TRCIDR7	ID Register 7
External	TRCIDR8	ID Register 8
External	TRCIDR9	ID Register 9
External	TRCIMSPEC0	IMP DEF Register 0
External	TRCIMSPEC<n>	IMP DEF Register <n>
External	TRCITEEDCR	Instrumentation Trace Extension External Debug Control Register
External	TRCPRGCTLR	Programming Control Register
External	TRCQCTLR	Q Element Control Register
External	TRCRSCTLR<n>	Resource Selection Control Register <n>
External	TRCRSR	Resources Status Register

Exec state	Name	Description
External	TRCSEQEVR<n>	Sequencer State Transition Control Register <n>
External	TRCSEORSTEVR	Sequencer Reset Control Register
External	TRCSEQSTR	Sequencer State Register
External	TRCSSCCR<n>	Single-shot Comparator Control Register <n>
External	TRCSSCSR<n>	Single-shot Comparator Control Status Register <n>
External	TRCSSPCICR<n>	Single-shot Processing Element Comparator Input Control Register <n>
External	TRCSTALLCTLR	Stall Control Register
External	TRCSTATR	Trace Status Register
External	TRCSYNCPR	Synchronization Period Register
External	TRCTRACEIDR	Trace ID Register
External	TRCTSCTLR	Timestamp Control Register
External	TRCVICTLR	ViewInst Main Control Register
External	TRCVIECTLR	ViewInst Include/Exclude Control Register
External	TRCVIPCSSCTLR	ViewInst Start/Stop PE Comparator Control Register
External	TRCVISSCTLR	ViewInst Start/Stop Control Register
External	TRCVMIDCCTLR0	Virtual Context Identifier Comparator Control Register 0
External	TRCVMIDCCTLR1	Virtual Context Identifier Comparator Control Register 1
External	TRCVMIDCVR<n>	Virtual Context Identifier Comparator Value Register <n>

In the GIC functional group:

Exec state	Name	Description
AArch32	ICC_AP0R<n>	Interrupt Controller Active Priorities Group 0 Registers
AArch32	ICC_AP1R<n>	Interrupt Controller Active Priorities Group 1 Registers
AArch32	ICC_ASGI1R	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
AArch32	ICC_BPR0	Interrupt Controller Binary Point Register 0
AArch32	ICC_BPR1	Interrupt Controller Binary Point Register 1
AArch32	ICC_CTLR	Interrupt Controller Control Register
AArch32	ICC_DIR	Interrupt Controller Deactivate Interrupt Register
AArch32	ICC_EOIR0	Interrupt Controller End Of Interrupt Register 0
AArch32	ICC_EOIR1	Interrupt Controller End Of Interrupt Register 1
AArch32	ICC_HPPIR0	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch32	ICC_HPPIR1	Interrupt Controller Highest Priority Pending Interrupt Register 1
AArch32	ICC_HSRE	Interrupt Controller Hyp System Register Enable register
AArch32	ICC_IAR0	Interrupt Controller Interrupt Acknowledge Register 0
AArch32	ICC_IAR1	Interrupt Controller Interrupt Acknowledge Register 1
AArch32	ICC_IGRPEN0	Interrupt Controller Interrupt Group 0 Enable register
AArch32	ICC_IGRPEN1	Interrupt Controller Interrupt Group 1 Enable register
AArch32	ICC_MCTLR	Interrupt Controller Monitor Control Register
AArch32	ICC_MGRPEN1	Interrupt Controller Monitor Interrupt Group 1 Enable register
AArch32	ICC_MSRE	Interrupt Controller Monitor System Register Enable register
AArch32	ICC_PMR	Interrupt Controller Interrupt Priority Mask Register
AArch32	ICC_RPR	Interrupt Controller Running Priority Register
AArch32	ICC_SGI0R	Interrupt Controller Software Generated Interrupt Group 0 Register
AArch32	ICC_SGI1R	Interrupt Controller Software Generated Interrupt Group 1 Register
AArch32	ICC_SRE	Interrupt Controller System Register Enable register
AArch32	ICH_AP0R<n>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch32	ICH_AP1R<n>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch32	ICH_EISR	Interrupt Controller End of Interrupt Status Register
AArch32	ICH_ELRSR	Interrupt Controller Empty List Register Status Register
AArch32	ICH_HCR	Interrupt Controller Hyp Control Register
AArch32	ICH_LR<n>	Interrupt Controller List Registers
AArch32	ICH_LRC<n>	Interrupt Controller List Registers
AArch32	ICH_MISR	Interrupt Controller Maintenance Interrupt State Register
AArch32	ICH_VMCR	Interrupt Controller Virtual Machine Control Register
AArch32	ICH_VTR	Interrupt Controller VGIC Type Register
AArch32	ICV_AP0R<n>	Interrupt Controller Virtual Active Priorities Group 0 Registers
AArch32	ICV_AP1R<n>	Interrupt Controller Virtual Active Priorities Group 1 Registers
AArch32	ICV_BPR0	Interrupt Controller Virtual Binary Point Register 0
AArch32	ICV_BPR1	Interrupt Controller Virtual Binary Point Register 1
AArch32	ICV_CTLR	Interrupt Controller Virtual Control Register
AArch32	ICV_DIR	Interrupt Controller Deactivate Virtual Interrupt Register
AArch32	ICV_EOIR0	Interrupt Controller Virtual End Of Interrupt Register 0

Exec state	Name	Description
AArch32	ICV EOIR1	Interrupt Controller Virtual End Of Interrupt Register 1
AArch32	ICV HPPIR0	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch32	ICV HPPIR1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
AArch32	ICV IAR0	Interrupt Controller Virtual Interrupt Acknowledge Register 0
AArch32	ICV IAR1	Interrupt Controller Virtual Interrupt Acknowledge Register 1
AArch32	ICV IGRPEN0	Interrupt Controller Virtual Interrupt Group 0 Enable register
AArch32	ICV IGRPEN1	Interrupt Controller Virtual Interrupt Group 1 Enable register
AArch32	ICV PMR	Interrupt Controller Virtual Interrupt Priority Mask Register
AArch32	ICV RPR	Interrupt Controller Virtual Running Priority Register
AArch64	ICC AP0R<n> EL1	Interrupt Controller Active Priorities Group 0 Registers
AArch64	ICC AP1R<n> EL1	Interrupt Controller Active Priorities Group 1 Registers
AArch64	ICC ASGI1R EL1	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
AArch64	ICC BPR0 EL1	Interrupt Controller Binary Point Register 0
AArch64	ICC BPR1 EL1	Interrupt Controller Binary Point Register 1
AArch64	ICC CTLR EL1	Interrupt Controller Control Register (EL1)
AArch64	ICC CTLR EL3	Interrupt Controller Control Register (EL3)
AArch64	ICC DIR EL1	Interrupt Controller Deactivate Interrupt Register
AArch64	ICC EOIR0 EL1	Interrupt Controller End Of Interrupt Register 0
AArch64	ICC EOIR1 EL1	Interrupt Controller End Of Interrupt Register 1
AArch64	ICC HPPIR0 EL1	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch64	ICC HPPIR1 EL1	Interrupt Controller Highest Priority Pending Interrupt Register 1
AArch64	ICC IAR0 EL1	Interrupt Controller Interrupt Acknowledge Register 0
AArch64	ICC IAR1 EL1	Interrupt Controller Interrupt Acknowledge Register 1
AArch64	ICC IGRPEN0 EL1	Interrupt Controller Interrupt Group 0 Enable register
AArch64	ICC IGRPEN1 EL1	Interrupt Controller Interrupt Group 1 Enable register
AArch64	ICC IGRPEN1 EL3	Interrupt Controller Interrupt Group 1 Enable register (EL3)
AArch64	ICC NMIA1R1 EL1	Interrupt Controller Non-maskable Interrupt Acknowledge Register 1
AArch64	ICC PMR EL1	Interrupt Controller Interrupt Priority Mask Register
AArch64	ICC RPR EL1	Interrupt Controller Running Priority Register
AArch64	ICC SGI0R EL1	Interrupt Controller Software Generated Interrupt Group 0 Register
AArch64	ICC SGI1R EL1	Interrupt Controller Software Generated Interrupt Group 1 Register
AArch64	ICC SRE EL1	Interrupt Controller System Register Enable register (EL1)
AArch64	ICC SRE EL2	Interrupt Controller System Register Enable register (EL2)
AArch64	ICC SRE EL3	Interrupt Controller System Register Enable register (EL3)
AArch64	ICH AP0R<n> EL2	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch64	ICH AP1R<n> EL2	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch64	ICH EISR EL2	Interrupt Controller End of Interrupt Status Register
AArch64	ICH ELRSR EL2	Interrupt Controller Empty List Register Status Register
AArch64	ICH HCR EL2	Interrupt Controller Hyp Control Register
AArch64	ICH LR<n> EL2	Interrupt Controller List Registers
AArch64	ICH MISR EL2	Interrupt Controller Maintenance Interrupt State Register
AArch64	ICH VMCR EL2	Interrupt Controller Virtual Machine Control Register
AArch64	ICH VTR EL2	Interrupt Controller VGIC Type Register
AArch64	ICV AP0R<n> EL1	Interrupt Controller Virtual Active Priorities Group 0 Registers
AArch64	ICV AP1R<n> EL1	Interrupt Controller Virtual Active Priorities Group 1 Registers
AArch64	ICV BPR0 EL1	Interrupt Controller Virtual Binary Point Register 0
AArch64	ICV BPR1 EL1	Interrupt Controller Virtual Binary Point Register 1
AArch64	ICV CTLR EL1	Interrupt Controller Virtual Control Register
AArch64	ICV DIR EL1	Interrupt Controller Deactivate Virtual Interrupt Register
AArch64	ICV EOIR0 EL1	Interrupt Controller Virtual End Of Interrupt Register 0
AArch64	ICV EOIR1 EL1	Interrupt Controller Virtual End Of Interrupt Register 1
AArch64	ICV HPPIR0 EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch64	ICV HPPIR1 EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
AArch64	ICV IAR0 EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 0
AArch64	ICV IAR1 EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 1
AArch64	ICV IGRPEN0 EL1	Interrupt Controller Virtual Interrupt Group 0 Enable register
AArch64	ICV IGRPEN1 EL1	Interrupt Controller Virtual Interrupt Group 1 Enable register
AArch64	ICV NMIA1R1 EL1	Interrupt Controller Virtual Non-maskable Interrupt Acknowledge Register 1
AArch64	ICV PMR EL1	Interrupt Controller Virtual Interrupt Priority Mask Register
AArch64	ICV RPR EL1	Interrupt Controller Virtual Running Priority Register

In the GICD functional group:

Exec state	Name	Description
External	GICD_CLRSPI_NSR	Clear Non-secure SPI Pending Register
External	GICD_CLRSPI_SR	Clear Secure SPI Pending Register
External	GICD_CPENDSGIR<n>	SGI Clear-Pending Registers
External	GICD_CTLR	Distributor Control Register
External	GICD_ICACTIVER<n>	Interrupt Clear-Active Registers
External	GICD_ICACTIVER<n>E	Interrupt Clear-Active Registers (extended SPI range)
External	GICD_ICENABLER<n>	Interrupt Clear-Enable Registers
External	GICD_ICENABLER<n>E	Interrupt Clear-Enable Registers
External	GICD_ICFGR<n>	Interrupt Configuration Registers
External	GICD_ICFGR<n>E	Interrupt Configuration Registers (Extended SPI Range)
External	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers
External	GICD_ICPENDR<n>E	Interrupt Clear-Pending Registers (extended SPI range)
External	GICD_IGROUPR<n>	Interrupt Group Registers
External	GICD_IGROUPR<n>E	Interrupt Group Registers (extended SPI range)
External	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers
External	GICD_IGRPMODR<n>E	Interrupt Group Modifier Registers (extended SPI range)
External	GICD_IIDR	Distributor Implementer Identification Register
External	GICD_INMIR<n>	Non-maskable Interrupt Registers, x = 0 to 31
External	GICD_INMIR<n>E	Non-maskable Interrupt Registers for Extended SPIs, x = 0 to 31
External	GICD_IPRIORITYR<n>	Interrupt Priority Registers
External	GICD_IPRIORITYR<n>E	Holds the priority of the corresponding interrupt for each extended SPI supported by the GIC.
External	GICD_IROUTER<n>	Interrupt Routing Registers
External	GICD_IROUTER<n>E	Interrupt Routing Registers (Extended SPI Range)
External	GICD_ISACTIVER<n>	Interrupt Set-Active Registers
External	GICD_ISACTIVER<n>E	Interrupt Set-Active Registers (extended SPI range)
External	GICD_ISENABLER<n>	Interrupt Set-Enable Registers
External	GICD_ISENABLER<n>E	Interrupt Set-Enable Registers
External	GICD_ISPENDR<n>	Interrupt Set-Pending Registers
External	GICD_ISPENDR<n>E	Interrupt Set-Pending Registers (extended SPI range)
External	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers
External	GICD_NSACR<n>	Non-secure Access Control Registers
External	GICD_NSACR<n>E	Non-secure Access Control Registers
External	GICD_SETSPI_NSR	Set Non-secure SPI Pending Register
External	GICD_SETSPI_SR	Set Secure SPI Pending Register
External	GICD_SGIR	Software Generated Interrupt Register
External	GICD_SPENDSGIR<n>	SGI Set-Pending Registers
External	GICD_STATUSR	Error Reporting Status Register
External	GICD_TYPER	Interrupt Controller Type Register
External	GICD_TYPER2	Interrupt Controller Type Register 2
External	GICM_CLRSPI_NSR	Clear Non-secure SPI Pending Register
External	GICM_CLRSPI_SR	Clear Secure SPI Pending Register
External	GICM_IIDR	Distributor Implementer Identification Register
External	GICM_SETSPI_NSR	Set Non-secure SPI Pending Register
External	GICM_SETSPI_SR	Set Secure SPI Pending Register
External	GICM_TYPER	Distributor MSI Type Register

In the GICR functional group:

Exec state	Name	Description
External	GICR_CLRLPIR	Clear LPI Pending Register
External	GICR_CTLR	Redistributor Control Register
External	GICR_ICACTIVER0	Interrupt Clear-Active Register 0
External	GICR_ICACTIVER<n>E	Interrupt Clear-Active Registers
External	GICR_ICENABLER0	Interrupt Clear-Enable Register 0
External	GICR_ICENABLER<n>E	Interrupt Clear-Enable Registers
External	GICR_ICFGR0	Interrupt Configuration Register 0
External	GICR_ICFGR1	Interrupt Configuration Register 1
External	GICR_ICFGR<n>E	Interrupt configuration registers
External	GICR_ICPENDR0	Interrupt Clear-Pending Register 0

Exec state	Name	Description
External	GICR_ICPENDR<n>E	Interrupt Clear-Pending Registers
External	GICR_IGROUPR0	Interrupt Group Register 0
External	GICR_IGROUPR<n>E	Interrupt Group Registers
External	GICR_IGRPMODR0	Interrupt Group Modifier Register 0
External	GICR_IGRPMODR<n>E	Interrupt Group Modifier Registers
External	GICR_IIDR	Redistributor Implementer Identification Register
External	GICR_INMIRO	Non-maskable Interrupt Register for PPIs.
External	GICR_INMIR<n>E	Non-maskable Interrupt Registers for Extended PPIs, x = 1 to 2.
External	GICR_INVALLR	Redistributor Invalidate All Register
External	GICR_INVLPIR	Redistributor Invalidate LPI Register
External	GICR_IPRIORITYR<n>	Interrupt Priority Registers
External	GICR_IPRIORITYR<n>E	Interrupt Priority Registers (extended PPI range)
External	GICR_ISACTIVER0	Interrupt Set-Active Register 0
External	GICR_ISACTIVER<n>E	Interrupt Set-Active Registers
External	GICR_ISENBALER0	Interrupt Set-Enable Register 0
External	GICR_ISENBALER<n>E	Interrupt Set-Enable Registers
External	GICR_ISPENDR0	Interrupt Set-Pending Register 0
External	GICR_ISPENDR<n>E	Interrupt Set-Pending Registers
External	GICR_MPAMIDR	Report maximum PARTID and PMG Register
External	GICR_NSACR	Non-secure Access Control Register
External	GICR_PARTIDR	Set PARTID and PMG Register
External	GICR_PENDBASER	Redistributor LPI Pending Table Base Address Register
External	GICR_PROPBASER	Redistributor Properties Base Address Register
External	GICR_SETLPIR	Set LPI Pending Register
External	GICR_STATUSR	Error Reporting Status Register
External	GICR_SYNCR	Redistributor Synchronize Register
External	GICR_TYPER	Redistributor Type Register
External	GICR_VPENDBASER	Virtual Redistributor LPI Pending Table Base Address Register
External	GICR_VPROPBASER	Virtual Redistributor Properties Base Address Register
External	GICR_VSGIPENDR	Redistributor virtual SGI pending state register
External	GICR_VSGIR	Redistributor virtual SGI pending state request register
External	GICR_WAKER	Redistributor Wake Register

In the GICC functional group:

Exec state	Name	Description
External	GICC_ABPR	CPU Interface Aliased Binary Point Register
External	GICC_AEOIR	CPU Interface Aliased End Of Interrupt Register
External	GICC_AHPPIR	CPU Interface Aliased Highest Priority Pending Interrupt Register
External	GICC_AIAR	CPU Interface Aliased Interrupt Acknowledge Register
External	GICC_APR<n>	CPU Interface Active Priorities Registers
External	GICC_BPR	CPU Interface Binary Point Register
External	GICC_CTLR	CPU Interface Control Register
External	GICC_DIR	CPU Interface Deactivate Interrupt Register
External	GICC_EOIR	CPU Interface End Of Interrupt Register
External	GICC_HPPIR	CPU Interface Highest Priority Pending Interrupt Register
External	GICC_IAR	CPU Interface Interrupt Acknowledge Register
External	GICC_IIDR	CPU Interface Identification Register
External	GICC_NSAPR<n>	CPU Interface Non-secure Active Priorities Registers
External	GICC_PMR	CPU Interface Priority Mask Register
External	GICC_RPR	CPU Interface Running Priority Register
External	GICC_STATUSR	CPU Interface Status Register

In the GICV functional group:

Exec state	Name	Description
External	GICV_ABPR	Virtual Machine Aliased Binary Point Register
External	GICV_AEOIR	Virtual Machine Aliased End Of Interrupt Register
External	GICV_AHPPIR	Virtual Machine Aliased Highest Priority Pending Interrupt Register
External	GICV_AIAR	Virtual Machine Aliased Interrupt Acknowledge Register
External	GICV_APR<n>	Virtual Machine Active Priorities Registers
External	GICV_BPR	Virtual Machine Binary Point Register

Exec state	Name	Description
External	GICV_CTLR	Virtual Machine Control Register
External	GICV_DIR	Virtual Machine Deactivate Interrupt Register
External	GICV_EOIR	Virtual Machine End Of Interrupt Register
External	GICV_HPPIR	Virtual Machine Highest Priority Pending Interrupt Register
External	GICV_IAR	Virtual Machine Interrupt Acknowledge Register
External	GICV_IIDR	Virtual Machine CPU Interface Identification Register
External	GICV_PMR	Virtual Machine Priority Mask Register
External	GICV_RPR	Virtual Machine Running Priority Register
External	GICV_STATUSR	Virtual Machine Error Reporting Status Register

In the GICH functional group:

Exec state	Name	Description
External	GICH_APR<n>	Active Priorities Registers
External	GICH_EISR	End Interrupt Status Register
External	GICH_ELRSR	Empty List Register Status Register
External	GICH_HCR	Hypervisor Control Register
External	GICH_LR<n>	List Registers
External	GICH_MISR	Maintenance Interrupt Status Register
External	GICH_VMCR	Virtual Machine Control Register
External	GICH_VTR	Virtual Type Register

In the GITS functional group:

Exec state	Name	Description
External	GITS_BASER<n>	ITS Translation Table Descriptors
External	GITS_CBASER	ITS Command Queue Descriptor
External	GITS_CREADR	ITS Read Register
External	GITS_CTLR	ITS Control Register
External	GITS_CWRITER	ITS Write Register
External	GITS_IIDR	ITS Identification Register
External	GITS_MPAMIDR	Report maximum PARTID and PMG Register
External	GITS_MPIDR	Report ITS's affinity.
External	GITS_PARTIDR	Set PARTID and PMG Register
External	GITS_SGIR	ITS SGI Register
External	GITS_STATUSR	ITS Error Reporting Status Register
External	GITS_TRANSLATER	ITS Translation Register
External	GITS_TYPER	ITS Type Register
External	GITS_UMSIR	ITS Unmapped MSI register

In the RAS functional group:

Exec state	Name	Description
AArch32	DISR	Deferred Interrupt Status Register
AArch32	ERRIDR	Error Record ID Register
AArch32	ERRSELR	Error Record Select Register
AArch32	ERXADDR	Selected Error Record Address Register
AArch32	ERXADDR2	Selected Error Record Address Register 2
AArch32	ERXCTLR	Selected Error Record Control Register
AArch32	ERXCTLR2	Selected Error Record Control Register 2
AArch32	ERXFR	Selected Error Record Feature Register
AArch32	ERXFR2	Selected Error Record Feature Register 2
AArch32	ERXMISC0	Selected Error Record Miscellaneous Register 0
AArch32	ERXMISC1	Selected Error Record Miscellaneous Register 1
AArch32	ERXMISC2	Selected Error Record Miscellaneous Register 2
AArch32	ERXMISC3	Selected Error Record Miscellaneous Register 3
AArch32	ERXMISC4	Selected Error Record Miscellaneous Register 4
AArch32	ERXMISC5	Selected Error Record Miscellaneous Register 5
AArch32	ERXMISC6	Selected Error Record Miscellaneous Register 6
AArch32	ERXMISC7	Selected Error Record Miscellaneous Register 7
AArch32	ERXSTATUS	Selected Error Record Primary Status Register

Exec state	Name	Description
AArch32	VDFSR	Virtual SError Exception Syndrome Register
AArch32	VDISR	Virtual Deferred Interrupt Status Register
AArch64	DISR_EL1	Deferred Interrupt Status Register
AArch64	ERRIDR_EL1	Error Record ID Register
AArch64	ERRSELR_EL1	Error Record Select Register
AArch64	ERXADDR_EL1	Selected Error Record Address Register
AArch64	ERXCTLR_EL1	Selected Error Record Control Register
AArch64	ERXFR_EL1	Selected Error Record Feature Register
AArch64	ERXMISC0_EL1	Selected Error Record Miscellaneous Register 0
AArch64	ERXMISC1_EL1	Selected Error Record Miscellaneous Register 1
AArch64	ERXMISC2_EL1	Selected Error Record Miscellaneous Register 2
AArch64	ERXMISC3_EL1	Selected Error Record Miscellaneous Register 3
AArch64	ERXPFGCDN_EL1	Selected Pseudo-fault Generation Countdown register
AArch64	ERXPFGCTL_EL1	Selected Pseudo-fault Generation Control register
AArch64	ERXPFGF_EL1	Selected Pseudo-fault Generation Feature register
AArch64	ERXSTATUS_EL1	Selected Error Record Primary Status Register
AArch64	VDISR_EL2	Virtual Deferred Interrupt Status Register
AArch64	VSESR_EL2	Virtual SError Exception Syndrome Register
External	ERR<n>ADDR	Error Record <n> Address Register
External	ERR<n>CTLR	Error Record <n> Control Register
External	ERR<n>FR	Error Record <n> Feature Register
External	ERR<n>MISC0	Error Record <n> Miscellaneous Register 0
External	ERR<n>MISC1	Error Record <n> Miscellaneous Register 1
External	ERR<n>MISC2	Error Record <n> Miscellaneous Register 2
External	ERR<n>MISC3	Error Record <n> Miscellaneous Register 3
External	ERR<n>PFGCDN	Error Record <n> Pseudo-fault Generation Countdown Register
External	ERR<n>PFGCTL	Error Record <n> Pseudo-fault Generation Control Register
External	ERR<n>PFGF	Error Record <n> Pseudo-fault Generation Feature Register
External	ERR<n>STATUS	Error Record <n> Primary Status Register
External	ERRCIDR0	Component Identification Register 0
External	ERRCIDR1	Component Identification Register 1
External	ERRCIDR2	Component Identification Register 2
External	ERRCIDR3	Component Identification Register 3
External	ERRCRICR0	Critical Error Interrupt Configuration Register 0
External	ERRCRICR1	Critical Error Interrupt Configuration Register 1
External	ERRCRICR2	Critical Error Interrupt Configuration Register 2
External	ERRDEVAFF	Device Affinity Register
External	ERRDEVARCH	Device Architecture Register
External	ERRDEVID	Device Configuration Register
External	ERRERICR0	Error Recovery Interrupt Configuration Register 0
External	ERRERICR1	Error Recovery Interrupt Configuration Register 1
External	ERRERICR2	Error Recovery Interrupt Configuration Register 2
External	ERRFHICR0	Fault Handling Interrupt Configuration Register 0
External	ERRFHICR1	Fault Handling Interrupt Configuration Register 1
External	ERRFHICR2	Fault Handling Interrupt Configuration Register 2
External	ERRGSR	Error Group Status Register
External	ERRIIDR	Implementation Identification Register
External	ERRIMPEDEF<n>	IMPLEMENTATION DEFINED Register <n>
External	ERRIRQCR<n>	Generic Error Interrupt Configuration Register <n>
External	ERRIRQSR	Error Interrupt Status Register
External	ERRPIDR0	Peripheral Identification Register 0
External	ERRPIDR1	Peripheral Identification Register 1
External	ERRPIDR2	Peripheral Identification Register 2
External	ERRPIDR3	Peripheral Identification Register 3
External	ERRPIDR4	Peripheral Identification Register 4

In the MPAM functional group:

Exec state	Name	Description
AArch64	MPAM0_EL1	MPAM0 Register (EL1)
AArch64	MPAM1_EL1	MPAM1 Register (EL1)
AArch64	MPAM2_EL2	MPAM2 Register (EL2)

Exec state	Name	Description
AArch64	MPAM3_EL3	MPAM3 Register (EL3)
AArch64	MPAMHCR_EL2	MPAM Hypervisor Control Register (EL2)
AArch64	MPAMSM_EL1	MPAM Streaming Mode Register
AArch64	MPAMVPM0_EL2	MPAM Virtual PARTID Mapping Register 0
AArch64	MPAMVPM1_EL2	MPAM Virtual PARTID Mapping Register 1
AArch64	MPAMVPM2_EL2	MPAM Virtual PARTID Mapping Register 2
AArch64	MPAMVPM3_EL2	MPAM Virtual PARTID Mapping Register 3
AArch64	MPAMVPM4_EL2	MPAM Virtual PARTID Mapping Register 4
AArch64	MPAMVPM5_EL2	MPAM Virtual PARTID Mapping Register 5
AArch64	MPAMVPM6_EL2	MPAM Virtual PARTID Mapping Register 6
AArch64	MPAMVPM7_EL2	MPAM Virtual PARTID Mapping Register 7
AArch64	MPAMVPMV_EL2	MPAM Virtual Partition Mapping Valid Register
External	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register
External	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register
External	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register
External	MPAMCFG_CPB<n>	MPAM Cache Portion Bitmap Partition Configuration Register
External	MPAMCFG_DIS	MPAM Partition Configuration Disable Register
External	MPAMCFG_EN	MPAM Partition Configuration Enable Register
External	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register
External	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register
External	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register
External	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register
External	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register
External	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register
External	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register
External	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register
External	MPAMCFG_PRI	MPAM Priority Partition Configuration Register
External	MPAMF_AIDR	MPAM Architecture Identification Register
External	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register
External	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register
External	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register
External	MPAMF_ECR	MPAM Error Control Register
External	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register
External	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register
External	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register
External	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register
External	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register
External	MPAMF_ESR	MPAM Error Status Register
External	MPAMF_IDR	MPAM Features Identification Register
External	MPAMF_IIDR	MPAM Implementation Identification Register
External	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register
External	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register
External	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register
External	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register
External	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register
External	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register
External	MPAMF_SIDR	MPAM Features Secure Identification Register
External	MSMON_CAPT_EVT	MPAM Capture Event Generation Register
External	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register
External	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register
External	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register
External	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register
External	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register
External	MSMON_CSU	MPAM Cache Storage Usage Monitor Register

Exec state	Name	Description
External	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register
External	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register
External	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register
External	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register
External	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register
External	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register
External	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register
External	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register
External	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register
External	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register
External	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register
External	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register
External	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register

In the Pointer authentication functional group:

Exec state	Name	Description
AArch64	APDAKeyHi_EL1	Pointer Authentication Key A for Data (bits[127:64])
AArch64	APDAKeyLo_EL1	Pointer Authentication Key A for Data (bits[63:0])
AArch64	APDBKeyHi_EL1	Pointer Authentication Key B for Data (bits[127:64])
AArch64	APDBKeyLo_EL1	Pointer Authentication Key B for Data (bits[63:0])
AArch64	APGAKeyHi_EL1	Pointer Authentication Key A for Code (bits[127:64])
AArch64	APGAKeyLo_EL1	Pointer Authentication Key A for Code (bits[63:0])
AArch64	APIAKeyHi_EL1	Pointer Authentication Key A for Instruction (bits[127:64])
AArch64	APIAKeyLo_EL1	Pointer Authentication Key A for Instruction (bits[63:0])
AArch64	APIBKeyHi_EL1	Pointer Authentication Key B for Instruction (bits[127:64])
AArch64	APIBKeyLo_EL1	Pointer Authentication Key B for Instruction (bits[63:0])

In the AMU functional group:

Exec state	Name	Description
AArch32	AMCFGR	Activity Monitors Configuration Register
AArch32	AMCGCR	Activity Monitors Counter Group Configuration Register
AArch32	AMCNTENCLR0	Activity Monitors Count Enable Clear Register 0
AArch32	AMCNTENCLR1	Activity Monitors Count Enable Clear Register 1
AArch32	AMCNTENSET0	Activity Monitors Count Enable Set Register 0
AArch32	AMCNTENSET1	Activity Monitors Count Enable Set Register 1
AArch32	AMCR	Activity Monitors Control Register
AArch32	AMEVCNTR0<n>	Activity Monitors Event Counter Registers 0
AArch32	AMEVCNTR1<n>	Activity Monitors Event Counter Registers 1
AArch32	AMEVTYPER0<n>	Activity Monitors Event Type Registers 0
AArch32	AMEVTYPER1<n>	Activity Monitors Event Type Registers 1
AArch32	AMUSERENR	Activity Monitors User Enable Register
AArch64	AMCFGR_EL0	Activity Monitors Configuration Register
AArch64	AMCG1IDR_EL0	Activity Monitors Counter Group 1 Identification Register
AArch64	AMCGCR_EL0	Activity Monitors Counter Group Configuration Register
AArch64	AMCNTENCLR0_EL0	Activity Monitors Count Enable Clear Register 0
AArch64	AMCNTENCLR1_EL0	Activity Monitors Count Enable Clear Register 1
AArch64	AMCNTENSET0_EL0	Activity Monitors Count Enable Set Register 0
AArch64	AMCNTENSET1_EL0	Activity Monitors Count Enable Set Register 1
AArch64	AMCR_EL0	Activity Monitors Control Register
AArch64	AMEVCNTR0<n>_EL0	Activity Monitors Event Counter Registers 0
AArch64	AMEVCNTR1<n>_EL0	Activity Monitors Event Counter Registers 1
AArch64	AMEVCNTVOFF0<n>_EL2	Activity Monitors Event Counter Virtual Offset Registers 0
AArch64	AMEVCNTVOFF1<n>_EL2	Activity Monitors Event Counter Virtual Offset Registers 1
AArch64	AMEVTYPER0<n>_EL0	Activity Monitors Event Type Registers 0
AArch64	AMEVTYPER1<n>_EL0	Activity Monitors Event Type Registers 1
AArch64	AMUSERENR_EL0	Activity Monitors User Enable Register
External	AMCFGR	Activity Monitors Configuration Register
External	AMCGCR	Activity Monitors Counter Group Configuration Register
External	AMCIDR0	Activity Monitors Component Identification Register 0

Exec state	Name	Description
External	AMCIDR1	Activity Monitors Component Identification Register 1
External	AMCIDR2	Activity Monitors Component Identification Register 2
External	AMCIDR3	Activity Monitors Component Identification Register 3
External	AMCNTENCLR0	Activity Monitors Count Enable Clear Register 0
External	AMCNTENCLR1	Activity Monitors Count Enable Clear Register 1
External	AMCNTENSET0	Activity Monitors Count Enable Set Register 0
External	AMCNTENSET1	Activity Monitors Count Enable Set Register 1
External	AMCR	Activity Monitors Control Register
External	AMDEVAFF0	Activity Monitors Device Affinity Register 0
External	AMDEVAFF1	Activity Monitors Device Affinity Register 1
External	AMDEVARCH	Activity Monitors Device Architecture Register
External	AMDEVTYPE	Activity Monitors Device Type Register
External	AMEVCNTR0<n>	Activity Monitors Event Counter Registers 0
External	AMEVCNTR1<n>	Activity Monitors Event Counter Registers 1
External	AMEVTYPER0<n>	Activity Monitors Event Type Registers 0
External	AMEVTYPER1<n>	Activity Monitors Event Type Registers 1
External	AMIIDR	Activity Monitors Implementation Identification Register
External	AMPIDR0	Activity Monitors Peripheral Identification Register 0
External	AMPIDR1	Activity Monitors Peripheral Identification Register 1
External	AMPIDR2	Activity Monitors Peripheral Identification Register 2
External	AMPIDR3	Activity Monitors Peripheral Identification Register 3
External	AMPIDR4	Activity Monitors Peripheral Identification Register 4

In the Root functional group:

Exec state	Name	Description
AArch64	GPPCCR_EL3	Granule Protection Check Control Register (EL3)
AArch64	GPTBR_EL3	Granule Protection Table Base Register
AArch64	MFAAR_EL3	PhysicalPA Fault Address Register (EL3)

In the GIC ITS registers functional group:

Exec state	Name	Description
External	GITS_BASER<n>	ITS Translation Table Descriptors
External	GITS_CBASER	ITS Command Queue Descriptor
External	GITS_CREADR	ITS Read Register
External	GITS_CTLR	ITS Control Register
External	GITS_CWRITER	ITS Write Register
External	GITS_IIDR	ITS Identification Register
External	GITS_MPAMIDR	Report maximum PARTID and PMG Register
External	GITS_MPIDR	Report ITS's affinity.
External	GITS_PARTIDR	Set PARTID and PMG Register
External	GITS_SGIR	ITS SGI Register
External	GITS_STATUSR	ITS Error Reporting Status Register
External	GITS_TRANSLATER	ITS Translation Register
External	GITS_TYPER	ITS Type Register
External	GITS_UMSIR	ITS Unmapped MSI register

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

External registers

AMCFGR: Activity Monitors Configuration Register

AMCGCR: Activity Monitors Counter Group Configuration Register

AMCIDR0: Activity Monitors Component Identification Register 0

AMCIDR1: Activity Monitors Component Identification Register 1

AMCIDR2: Activity Monitors Component Identification Register 2

AMCIDR3: Activity Monitors Component Identification Register 3

AMCNTENCLR0: Activity Monitors Count Enable Clear Register 0

AMCNTENCLR1: Activity Monitors Count Enable Clear Register 1

AMCNTENSET0: Activity Monitors Count Enable Set Register 0

AMCNTENSET1: Activity Monitors Count Enable Set Register 1

AMCR: Activity Monitors Control Register

AMDEVAFF0: Activity Monitors Device Affinity Register 0

AMDEVAFF1: Activity Monitors Device Affinity Register 1

AMDEVARCH: Activity Monitors Device Architecture Register

AMDEVTYPE: Activity Monitors Device Type Register

AMEVCNTR0<n>: Activity Monitors Event Counter Registers 0

AMEVCNTR1<n>: Activity Monitors Event Counter Registers 1

AMEVTYPER0<n>: Activity Monitors Event Type Registers 0

AMEVTYPER1<n>: Activity Monitors Event Type Registers 1

AMIIDR: Activity Monitors Implementation Identification Register

AMPIDR0: Activity Monitors Peripheral Identification Register 0

AMPIDR1: Activity Monitors Peripheral Identification Register 1

AMPIDR2: Activity Monitors Peripheral Identification Register 2

AMPIDR3: Activity Monitors Peripheral Identification Register 3

AMPIDR4: Activity Monitors Peripheral Identification Register 4

ASICCTL: CTI External Multiplexer Control register

CNTACR<n>: Counter-timer Access Control Registers

CNTCR: Counter Control Register

CNTCV: Counter Count Value register

CNTEL0ACR: Counter-timer EL0 Access Control Register

CNTFID0: Counter Frequency ID

CNTFID<n>: Counter Frequency IDs, n > 0

CNTFRQ: Counter-timer Frequency

CNTID: Counter Identification Register

CNTNSAR: Counter-timer Non-secure Access Register

CNTPCT: Counter-timer Physical Count

CNTP_CTL: Counter-timer Physical Timer Control

CNTP_CVAL: Counter-timer Physical Timer CompareValue

CNTP_TVAL: Counter-timer Physical Timer TimerValue

CNTSCR: Counter Scale Register

CNTSR: Counter Status Register

CNTTIDR: Counter-timer Timer ID Register

CNTVCT: Counter-timer Virtual Count

CNTVOFF: Counter-timer Virtual Offset

CNTVOFF<n>: Counter-timer Virtual Offsets

CNTV_CTL: Counter-timer Virtual Timer Control

CNTV_CVAL: Counter-timer Virtual Timer CompareValue

CNTV_TVAL: Counter-timer Virtual Timer TimerValue

CounterID<n>: Counter ID registers

CTIAPPCLEAR: CTI Application Trigger Clear register

CTIAPPULSE: CTI Application Pulse register

CTIAPPSET: CTI Application Trigger Set register

CTIAUTHSTATUS: CTI Authentication Status register

CTICHINSTATUS: CTI Channel In Status register

CTICHOUTSTATUS: CTI Channel Out Status register

CTICIDR0: CTI Component Identification Register 0

CTICIDR1: CTI Component Identification Register 1

CTICIDR2: CTI Component Identification Register 2

CTICIDR3: CTI Component Identification Register 3

CTICLAIMCLR: CTI CLAIM Tag Clear register

CTICLAIMSET: CTI CLAIM Tag Set register

CTICONTROL: CTI Control register

CTIDEVAFF0: CTI Device Affinity register 0

CTIDEVAFF1: CTI Device Affinity register 1

CTIDEVARCH: CTI Device Architecture register

CTIDEVCTL: CTI Device Control register

CTIDEVID: CTI Device ID register 0

CTIDEVID1: CTI Device ID register 1

CTIDEVID2: CTI Device ID register 2

CTIDEVTYPE: CTI Device Type register

CTIGATE: CTI Channel Gate Enable register

CTIINEN<n>: CTI Input Trigger to Output Channel Enable registers

CTIINTACK: CTI Output Trigger Acknowledge register

CTIITCTRL: CTI Integration mode Control register

CTILAR: CTI Lock Access Register

CTILSR: CTI Lock Status Register

CTIOUTEN<n>: CTI Input Channel to Output Trigger Enable registers

CTIPIDR0: CTI Peripheral Identification Register 0

CTIPIDR1: CTI Peripheral Identification Register 1

CTIPIDR2: CTI Peripheral Identification Register 2

CTIPIDR3: CTI Peripheral Identification Register 3

CTIPIDR4: CTI Peripheral Identification Register 4

CTITRIGINSTATUS: CTI Trigger In Status register

CTITRIGOUTSTATUS: CTI Trigger Out Status register

[DBGAUTHSTATUS_EL1](#): Debug Authentication Status register

[DBGBCR<n>_EL1](#): Debug Breakpoint Control Registers

DBGBVR<n>_EL1: Debug Breakpoint Value Registers

DBGCLAIMCLR_EL1: Debug CLAIM Tag Clear register

DBGCLAIMSET_EL1: Debug CLAIM Tag Set register

DBGDTRRX_EL0: Debug Data Transfer Register, Receive

DBGDTRTX_EL0: Debug Data Transfer Register, Transmit

[DBGWCR<n>_EL1](#): Debug Watchpoint Control Registers

DBGWVR<n>_EL1: Debug Watchpoint Value Registers

EDAA32PFR: External Debug Auxiliary Processor Feature Register

EDACR: External Debug Auxiliary Control Register

EDCIDR0: External Debug Component Identification Register 0

EDCIDR1: External Debug Component Identification Register 1

EDCIDR2: External Debug Component Identification Register 2

EDCIDR3: External Debug Component Identification Register 3

EDCIDSr: External Debug Context ID Sample Register

EDDEVAFF0: External Debug Device Affinity register 0

EDDEVAFF1: External Debug Device Affinity register 1

[EDDEVARCH](#): External Debug Device Architecture register

[EDDEVID](#): External Debug Device ID register 0

[EDDEVID1](#): External Debug Device ID register 1

EDDEVID2: External Debug Device ID register 2

EDDEVTYPE: External Debug Device Type register

[EDDFR](#): External Debug Feature Register

[EDDFR1](#): External Debug Feature Register 1

EDECCR: External Debug Exception Catch Control Register

[EDECR](#): External Debug Execution Control Register

EDESR: External Debug Event Status Register

[EDHSR](#): External Debug ~~HaltingHalt~~ ~~Syndrom~~~~Status~~ Register

EDITCTRL: External Debug Integration mode Control register

EDITR: External Debug Instruction Transfer Register

EDLAR: External Debug Lock Access Register

EDLSR: External Debug Lock Status Register

[EDPCSR](#): External Debug Program Counter Sample Register

EDPFR: External Debug Processor Feature Register

EDPIDR0: External Debug Peripheral Identification Register 0

EDPIDR1: External Debug Peripheral Identification Register 1

EDPIDR2: External Debug Peripheral Identification Register 2

EDPIDR3: External Debug Peripheral Identification Register 3

EDPIDR4: External Debug Peripheral Identification Register 4

EDPRCR: External Debug Power/Reset Control Register

EDPRSR: External Debug Processor Status Register

EDRCR: External Debug Reserve Control Register

[EDSCR](#): External Debug Status and Control Register

[EDSCR2](#): External Debug Status and Control Register 2

EDVIDSR: External Debug Virtual Context Sample Register

EDWAR: External Debug Watchpoint Address Register

ERR<n>ADDR: Error Record <n> Address Register

[ERR<n>CTLR](#): Error Record <n> Control Register

[ERR<n>FR](#): Error Record <n> Feature Register

ERR<n>MISC0: Error Record <n> Miscellaneous Register 0

ERR<n>MISC1: Error Record <n> Miscellaneous Register 1

ERR<n>MISC2: Error Record <n> Miscellaneous Register 2

ERR<n>MISC3: Error Record <n> Miscellaneous Register 3

ERR<n>PFGCDN: Error Record <n> Pseudo-fault Generation Countdown Register

ERR<n>PFGCTL: Error Record <n> Pseudo-fault Generation Control Register

ERR<n>PFGF: Error Record <n> Pseudo-fault Generation Feature Register

[ERR<n>STATUS](#): Error Record <n> Primary Status Register

ERRCIDR0: Component Identification Register 0

ERRCIDR1: Component Identification Register 1

ERRCIDR2: Component Identification Register 2

ERRCIDR3: Component Identification Register 3

ERRCRICR0: Critical Error Interrupt Configuration Register 0

ERRCRICR1: Critical Error Interrupt Configuration Register 1

ERRCRICR2: Critical Error Interrupt Configuration Register 2

ERRDEVAFF: Device Affinity Register

[ERRDEVARCH](#): Device Architecture Register

[ERRDEVID](#): Device Configuration Register

ERRERICR0: Error Recovery Interrupt Configuration Register 0

ERRERICR1: Error Recovery Interrupt Configuration Register 1

ERRERICR2: Error Recovery Interrupt Configuration Register 2

ERRFHICR0: Fault Handling Interrupt Configuration Register 0

ERRFHICR1: Fault Handling Interrupt Configuration Register 1

ERRFHICR2: Fault Handling Interrupt Configuration Register 2

ERRGSR: Error Group Status Register

ERRIIDR: Implementation Identification Register

ERRIMPDEF<n>: IMPLEMENTATION DEFINED Register <n>

ERRIRQCR<n>: Generic Error Interrupt Configuration Register <n>

ERRIRQSR: Error Interrupt Status Register

ERRPIDR0: Peripheral Identification Register 0

ERRPIDR1: Peripheral Identification Register 1

ERRPIDR2: Peripheral Identification Register 2

ERRPIDR3: Peripheral Identification Register 3

ERRPIDR4: Peripheral Identification Register 4

GICC_ABPR: CPU Interface Aliased Binary Point Register

GICC_AEOIR: CPU Interface Aliased End Of Interrupt Register

GICC_AHPPIR: CPU Interface Aliased Highest Priority Pending Interrupt Register

GICC_AIAR: CPU Interface Aliased Interrupt Acknowledge Register

GICC_APR<n>: CPU Interface Active Priorities Registers

GICC_BPR: CPU Interface Binary Point Register

GICC_CTLR: CPU Interface Control Register

GICC_DIR: CPU Interface Deactivate Interrupt Register

GICC_EOIR: CPU Interface End Of Interrupt Register

GICC_HPPIR: CPU Interface Highest Priority Pending Interrupt Register

GICC_IAR: CPU Interface Interrupt Acknowledge Register

GICC_IIDR: CPU Interface Identification Register

GICC_NSAPR<n>: CPU Interface Non-secure Active Priorities Registers

GICC_PMR: CPU Interface Priority Mask Register

GICC_RPR: CPU Interface Running Priority Register

GICC_STATUSR: CPU Interface Status Register

GICD_CLRSPI_NSR: Clear Non-secure SPI Pending Register

GICD_CLRSPI_SR: Clear Secure SPI Pending Register

GICD_CPENDSGIR<n>: SGI Clear-Pending Registers

GICD_CTLR: Distributor Control Register

GICD_ICACTIVER<n>: Interrupt Clear-Active Registers

GICD_ICACTIVER<n>E: Interrupt Clear-Active Registers (extended SPI range)

GICD_ICENABLER<n>: Interrupt Clear-Enable Registers

GICD_ICENABLER<n>E: Interrupt Clear-Enable Registers

GICD_ICFGR<n>: Interrupt Configuration Registers

GICD_ICFGR<n>E: Interrupt Configuration Registers (Extended SPI Range)

GICD_ICPENDR<n>: Interrupt Clear-Pending Registers

GICD_ICPENDR<n>E: Interrupt Clear-Pending Registers (extended SPI range)

GICD_IGROUPR<n>: Interrupt Group Registers

GICD_IGROUPR<n>E: Interrupt Group Registers (extended SPI range)

GICD_IGRPMDR<n>: Interrupt Group Modifier Registers

GICD_IGRPMDR<n>E: Interrupt Group Modifier Registers (extended SPI range)

GICD_IIDR: Distributor Implementer Identification Register

[GICD_INMIR<n>](#): Non-maskable Interrupt Registers, x = 0 to 31

[GICD_INMIR<n>E](#): Non-maskable Interrupt Registers for Extended SPIs, x = 0 to 31

GICD_IPRIORITYR<n>: Interrupt Priority Registers

GICD_IPRIORITYR<n>E: Holds the priority of the corresponding interrupt for each extended SPI supported by the GIC.

[GICD_IROUTER<n>](#): Interrupt Routing Registers

[GICD_IROUTER<n>E](#): Interrupt Routing Registers (Extended SPI Range)

GICD_ISACTIVER<n>: Interrupt Set-Active Registers

GICD_ISACTIVER<n>E: Interrupt Set-Active Registers (extended SPI range)

GICD_ISENABLER<n>: Interrupt Set-Enable Registers

GICD_ISENABLER<n>E: Interrupt Set-Enable Registers

GICD_ISPENDR<n>: Interrupt Set-Pending Registers

GICD_ISPENDR<n>E: Interrupt Set-Pending Registers (extended SPI range)

GICD_ITARGETSR<n>: Interrupt Processor Targets Registers

GICD_NSACR<n>: Non-secure Access Control Registers

GICD_NSACR<n>E: Non-secure Access Control Registers

GICD_SETSPI_NSR: Set Non-secure SPI Pending Register

GICD_SETSPI_SR: Set Secure SPI Pending Register

GICD_SGIR: Software Generated Interrupt Register

GICD_SPENDSGIR<n>: SGI Set-Pending Registers

GICD_STATUSR: Error Reporting Status Register

GICD_TYPER: Interrupt Controller Type Register

GICD_TYPER2: Interrupt Controller Type Register 2

GICH_APR<n>: Active Priorities Registers

GICH_EISR: End Interrupt Status Register

GICH_ELRSR: Empty List Register Status Register

GICH_HCR: Hypervisor Control Register

GICH_LR<n>: List Registers

GICH_MISR: Maintenance Interrupt Status Register

GICH_VMCR: Virtual Machine Control Register

GICH_VTR: Virtual Type Register

GICM_CLRSPI_NSR: Clear Non-secure SPI Pending Register

GICM_CLRSPI_SR: Clear Secure SPI Pending Register

GICM_IIDR: Distributor Implementer Identification Register

GICM_SETSPI_NSR: Set Non-secure SPI Pending Register

GICM_SETSPI_SR: Set Secure SPI Pending Register

GICM_TYPER: Distributor MSI Type Register

GICR_CLRLPIR: Clear LPI Pending Register

GICR_CTLR: Redistributor Control Register

GICR_ICACTIVER0: Interrupt Clear-Active Register 0

GICR_ICACTIVER<n>E: Interrupt Clear-Active Registers

GICR_ICENABLER0: Interrupt Clear-Enable Register 0

GICR_ICENABLER<n>E: Interrupt Clear-Enable Registers

GICR_ICFGR0: Interrupt Configuration Register 0

GICR_ICFGR1: Interrupt Configuration Register 1

GICR_ICFGR<n>E: Interrupt configuration registers

GICR_ICPENDR0: Interrupt Clear-Pending Register 0

GICR_ICPENDR<n>E: Interrupt Clear-Pending Registers

GICR_IGROUPR0: Interrupt Group Register 0

GICR_IGROUPR<n>E: Interrupt Group Registers

GICR_IGRPMODR0: Interrupt Group Modifier Register 0

GICR_IGRPMODR<n>E: Interrupt Group Modifier Registers

GICR_IIDR: Redistributor Implementer Identification Register

[GICR_INMIR0](#): Non-maskable Interrupt Register for PPIs.

[GICR_INMIR<n>E](#): Non-maskable Interrupt Registers for Extended PPIs, x = 1 to 2.

GICR_INVALLR: Redistributor Invalidate All Register

GICR_INVLPIR: Redistributor Invalidate LPI Register

GICR_IPRIORITYR<n>: Interrupt Priority Registers

GICR_IPRIORITYR<n>E: Interrupt Priority Registers (extended PPI range)

GICR_ISACTIVER0: Interrupt Set-Active Register 0

GICR_ISACTIVER<n>E: Interrupt Set-Active Registers

GICR_ISENBALER0: Interrupt Set-Enable Register 0

GICR_ISENBALER<n>E: Interrupt Set-Enable Registers

GICR_ISPENDR0: Interrupt Set-Pending Register 0

GICR_ISPENDR<n>E: Interrupt Set-Pending Registers

GICR_MPAMIDR: Report maximum PARTID and PMG Register

GICR_NSACR: Non-secure Access Control Register

GICR_PARTIDR: Set PARTID and PMG Register

GICR_PENDBASER: Redistributor LPI Pending Table Base Address Register

GICR_PROPBASER: Redistributor Properties Base Address Register

GICR_SETLPIR: Set LPI Pending Register

GICR_STATUSR: Error Reporting Status Register

GICR_SYNCR: Redistributor Synchronize Register

[GICR_TYPER](#): Redistributor Type Register

GICR_VPENDBASER: Virtual Redistributor LPI Pending Table Base Address Register

GICR_VPROPBASER: Virtual Redistributor Properties Base Address Register

GICR_VSGIPENDR: Redistributor virtual SGI pending state register

GICR_VSGIR: Redistributor virtual SGI pending state request register

GICR_WAKER: Redistributor Wake Register

GICV_ABPR: Virtual Machine Aliased Binary Point Register

GICV_AEOIR: Virtual Machine Aliased End Of Interrupt Register

GICV_AHPPIR: Virtual Machine Aliased Highest Priority Pending Interrupt Register

GICV_AIAR: Virtual Machine Aliased Interrupt Acknowledge Register

GICV_APR<n>: Virtual Machine Active Priorities Registers

GICV_BPR: Virtual Machine Binary Point Register

GICV_CTLR: Virtual Machine Control Register

GICV_DIR: Virtual Machine Deactivate Interrupt Register

GICV_EOIR: Virtual Machine End Of Interrupt Register

GICV_HPPIR: Virtual Machine Highest Priority Pending Interrupt Register

GICV_IAR: Virtual Machine Interrupt Acknowledge Register

GICV_IIDR: Virtual Machine CPU Interface Identification Register

GICV_PMR: Virtual Machine Priority Mask Register

GICV_RPR: Virtual Machine Running Priority Register

GICV_STATUSR: Virtual Machine Error Reporting Status Register

GITS_BASER<n>: ITS Translation Table Descriptors

GITS_CBASER: ITS Command Queue Descriptor

GITS_CREADR: ITS Read Register

GITS_CTLR: ITS Control Register

GITS_CWRITER: ITS Write Register

GITS_IIDR: ITS Identification Register

GITS_MPAMIDR: Report maximum PARTID and PMG Register

GITS_MPIDR: Report ITS's affinity.

GITS_PARTIDR: Set PARTID and PMG Register

GITS_SGIR: ITS SGI Register

GITS_STATUSR: ITS Error Reporting Status Register

GITS_TRANSLATER: ITS Translation Register

GITS_TYPER: ITS Type Register

GITS_UMSIR: ITS Unmapped MSI register

MIDR_EL1: Main ID Register

MPAMCFG_CASSOC: MPAM Cache Maximum Associativity Partition Configuration Register

MPAMCFG_CMAX: MPAM Cache Maximum Capacity Partition Configuration Register

MPAMCFG_CMIN: MPAM Cache Minimum Capacity Partition Configuration Register

MPAMCFG_CPBM<n>: MPAM Cache Portion Bitmap Partition Configuration Register

[MPAMCFG_DIS](#): MPAM Partition Configuration Disable Register

MPAMCFG_EN: MPAM Partition Configuration Enable Register

MPAMCFG_EN_FLAGS: MPAM Partition Configuration Enable Flags Register

MPAMCFG_INTPARTID: MPAM Internal PARTID Narrowing Configuration Register

MPAMCFG_MBW_MAX: MPAM Memory Bandwidth Maximum Partition Configuration Register

MPAMCFG_MBW_MIN: MPAM Memory Bandwidth Minimum Partition Configuration Register

MPAMCFG_MBW_PBM<n>: MPAM Bandwidth Portion Bitmap Partition Configuration Register

MPAMCFG_MBW_PROP: MPAM Memory Bandwidth Proportional Stride Partition Configuration Register

MPAMCFG_MBW_WINWD: MPAM Memory Bandwidth Partitioning Window Width Configuration Register

[MPAMCFG_PART_SEL](#): MPAM Partition Configuration Selection Register

MPAMCFG_PRI: MPAM Priority Partition Configuration Register

MPAMF_AIDR: MPAM Architecture Identification Register

MPAMF_CCAP_IDR: MPAM Features Cache Capacity Partitioning ID register

MPAMF_CPOR_IDR: MPAM Features Cache Portion Partitioning ID register

MPAMF_CSUMON_IDR: MPAM Features Cache Storage Usage Monitoring ID register

MPAMF_ECR: MPAM Error Control Register

[MPAMF_ERR_MSI_ADDR_H](#): MPAM Error MSI High-part Address Register

[MPAMF_ERR_MSI_ADDR_L](#): MPAM Error MSI Low-part Address Register

[MPAMF_ERR_MSI_ATTR](#): MPAM Error MSI Write Attributes Register

[MPAMF_ERR_MSI_DATA](#): MPAM Error MSI Data Register

[MPAMF_ERR_MSI_MPAM](#): MPAM Error MSI Write MPAM Information Register

MPAMF_ESR: MPAM Error Status Register

MPAMF_IDR: MPAM Features Identification Register

MPAMF_IIDR: MPAM Implementation Identification Register

MPAMF_IMPL_IDR: MPAM Implementation-Specific Partitioning Feature Identification Register

MPAMF_MBWUMON_IDR: MPAM Features Memory Bandwidth Usage Monitoring ID register

MPAMF_MBW_IDR: MPAM Memory Bandwidth Partitioning Identification Register

MPAMF_MSMON_IDR: MPAM Resource Monitoring Identification Register

MPAMF_PARTID_NRW_IDR: MPAM PARTID Narrowing ID register

MPAMF_PRI_IDR: MPAM Priority Partitioning Identification Register

MPAMF_SIDR: MPAM Features Secure Identification Register

[MSMON_CAPT_EVNT](#): MPAM Capture Event Generation Register

[MSMON_CFG_CSU_CTL](#): MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register

MSMON_CFG_CSU_FLT: MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register

[MSMON_CFG_MBWU_CTL](#): MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register

MSMON_CFG_MBWU_FLT: MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register

MSMON_CFG_MON_SEL: MPAM Monitor Instance Selection Register

MSMON_CSU: MPAM Cache Storage Usage Monitor Register

MSMON_CSU_CAPTURE: MPAM Cache Storage Usage Monitor Capture Register

[MSMON_CSU_OFSR](#): MPAM CSU Monitor Overflow Status Register

MSMON_MBWU: MPAM Memory Bandwidth Usage Monitor Register

MSMON_MBWU_CAPTURE: MPAM Memory Bandwidth Usage Monitor Capture Register

MSMON_MBWU_L: MPAM Long Memory Bandwidth Usage Monitor Register

MSMON_MBWU_L_CAPTURE: MPAM Long Memory Bandwidth Usage Monitor Capture Register

[MSMON_MBWU_OFSR](#): MPAM MBWU Monitor Overflow Status Register

MSMON_OFLOW_MSI_ADDR_H: MPAM Monitor Overflow MSI Write High-part Address Register

MSMON_OFLOW_MSI_ADDR_L: MPAM Monitor Overflow MSI Low-part Address Register

MSMON_OFLOW_MSI_ATTR: MPAM Monitor Overflow MSI Write Attributes Register

MSMON_OFLOW_MSI_DATA: MPAM Monitor Overflow MSI Write Data Register

MSMON_OFLOW_MSI_MPAM: MPAM Monitor Overflow MSI Write MPAM Information Register

[MSMON_OFLOW_SR](#): MPAM Monitor Overflow Status Register

OSLAR_EL1: OS Lock Access Register

[PMAUTHSTATUS](#): Performance Monitors Authentication Status register

[PMCCFILTR_EL0](#): Performance Monitors Cycle Counter Filter Register

[PMCCIDSR](#): CONTEXTIDR_ELx Sample Register

[PMCCNTR_EL0](#): Performance Monitors Cycle Counter

[PMCCNTSVR_EL1](#): Performance Monitors Cycle Count Saved Value Register

[PMCEID0](#): Performance Monitors Common Event Identification register 0

[PMCEID1](#): Performance Monitors Common Event Identification register 1

[PMCEID2](#): Performance Monitors Common Event Identification register 2

[PMCEID3](#): Performance Monitors Common Event Identification register 3

[PMCFGR](#): Performance Monitors Configuration Register

[PMCGCR0](#): Counter Group Configuration Register 0

[PMCID1SR](#): CONTEXTIDR_EL1 Sample Register

[PMCID2SR](#): CONTEXTIDR_EL2 Sample Register

[PMCIDR0](#): Performance Monitors Component Identification Register 0

[PMCIDR1](#): Performance Monitors Component Identification Register 1

[PMCIDR2](#): Performance Monitors Component Identification Register 2

[PMCIDR3](#): Performance Monitors Component Identification Register 3

[PMCNTEN](#): Performance Monitors Count Enable register

[PMCNTENCLR_EL0](#): Performance Monitors Count Enable Clear register

[PMCNTENSET_EL0](#): Performance Monitors Count Enable Set register

[PMCR_EL0](#): Performance Monitors Control Register

[PMDEVAFF](#): Performance Monitors Device Affinity register

[PMDEVAFF0](#): Performance Monitors Device Affinity register 0

[PMDEVAFF1](#): Performance Monitors Device Affinity register 1

[PMDEVARCH](#): Performance Monitors Device Architecture register

[PMDEVID](#): Performance Monitors Device ID register

[PMDEVTYPE](#): Performance Monitors Device Type register

[PMEVCNTR<n>_EL0](#): Performance Monitors Event Count Registers

[PMEVCNTSVR<n>_EL1](#): Performance Monitors Event Count Saved Value Register <n>

[PMEVTYPER<n>_EL0](#): Performance Monitors Event Type Registers

[PMICFILTR_EL0](#): Performance Monitors Instruction Counter Filter Register

[PMICNTR_EL0](#): Performance Monitors Instruction Counter Register

[PMICNTSVR_EL1](#): Performance Monitors Instruction Count Saved Value Register

[PMIIDR](#): Performance Monitors Implementation Identification Register

[PMINTEN](#): Performance Monitors Interrupt Enable register

[PMINTENCLR_EL1](#): Performance Monitors Interrupt Enable Clear register

[PMINTENSET_EL1](#): Performance Monitors Interrupt Enable Set register

[PMITCTRL](#): Performance Monitors Integration mode Control register

[PMLAR](#): Performance Monitors Lock Access Register

[PMLSR](#): Performance Monitors Lock Status Register

[PMMIR](#): Performance Monitors Machine Identification Register

[PMOVS](#): Performance Monitors Overflow Flag Status register

[PMOVSLR_EL0](#): Performance Monitors Overflow Flag Status Clear register

[PMOVSSET_EL0](#): Performance Monitors Overflow Flag Status Set register

[PMPCCTL](#): PC Sample-based Profiling Control Register

[PMPCSR](#): Program Counter Sample Register

[PMPIDR0](#): Performance Monitors Peripheral Identification Register 0

[PMPIDR1](#): Performance Monitors Peripheral Identification Register 1

[PMPIDR2](#): Performance Monitors Peripheral Identification Register 2

[PMPIDR3](#): Performance Monitors Peripheral Identification Register 3

[PMPIDR4](#): Performance Monitors Peripheral Identification Register 4

[PMSSCR_EL1](#): Performance Monitors Snapshot Status and Capture Register

[PMSWINC_EL0](#): Performance Monitors Software Increment register

[PMVCIDSR](#): CONTEXTIDR_EL1 and VMID Sample Register

[PMVIDSR](#): VMID Sample Register

[TRBAUTHSTATUS](#): Authentication Status Register

[TRBBASER_EL1](#): Trace Buffer Base Address Register

[TRBCIDR0](#): Component Identification Register 0

[TRBCIDR1](#): Component Identification Register 1

[TRBCIDR2](#): Component Identification Register 2

[TRBCIDR3](#): Component Identification Register 3

[TRBCR](#): Trace Buffer Control Register

[TRBDEVAFF](#): Device Affinity Register

[TRBDEVARCH](#): Trace Buffer Device Architecture Register
[TRBDEVID](#): Device Configuration Register
[TRBDEVID1](#): Device Configuration Register 1
[TRBDEVID2](#): Device Configuration Register 2
[TRBDEVTYPE](#): Device Type Register
[TRBIDR_EL1](#): Trace Buffer ID Register
[TRBITCTRL](#): Integration Mode Control Register
[TRBLAR](#): Lock Access Register
[TRBLIMITR_EL1](#): Trace Buffer Limit Address Register
[TRBLSR](#): Lock Status Register
[TRBMAR_EL1](#): Trace Buffer Memory Attribute Register
[TRBMPAM](#): Trace Buffer MPAM Configuration Register
[TRBPIDR0](#): Peripheral Identification Register 0
[TRBPIDR1](#): Peripheral Identification Register 1
[TRBPIDR2](#): Peripheral Identification Register 2
[TRBPIDR3](#): Peripheral Identification Register 3
[TRBPIDR4](#): Peripheral Identification Register 4
[TRBPIDR5](#): Peripheral Identification Register 5
[TRBPIDR6](#): Peripheral Identification Register 6
[TRBPIDR7](#): Peripheral Identification Register 7
[TRBPTR_EL1](#): Trace Buffer Write Pointer Register
[TRBSR_EL1](#): Trace Buffer Status/syndrome Register
[TRBTRG_EL1](#): Trace Buffer Trigger Counter Register
[TRCACATR<n>](#): Address Comparator Access Type Register <n>
[TRCACVR<n>](#): Address Comparator Value Register <n>
[TRCAUTHSTATUS](#): Authentication Status Register
[TRCAUXCTLR](#): Auxiliary Control Register
[TRCBBCTLR](#): Branch Broadcast Control Register
[TRCCCCTLR](#): Cycle Count Control Register
[TRCCIDCCTLR0](#): Context Identifier Comparator Control Register 0
[TRCCIDCCTLR1](#): Context Identifier Comparator Control Register 1
[TRCCIDCVR<n>](#): Context Identifier Comparator Value Registers <n>
[TRCCIDR0](#): Component Identification Register 0
[TRCCIDR1](#): Component Identification Register 1
[TRCCIDR2](#): Component Identification Register 2
[TRCCIDR3](#): Component Identification Register 3

[TRCCLAIMCLR](#): Claim Tag Clear Register

[TRCCLAIMSET](#): Claim Tag Set Register

[TRCCNTCTLR<n>](#): Counter Control Register <n>

[TRCCNTRLDVR<n>](#): Counter Reload Value Register <n>

[TRCCNTVR<n>](#): Counter Value Register <n>

[TRCCONFIGR](#): Trace Configuration Register

[TRCDEVAFE](#): Device Affinity Register

[TRCDEVARCH](#): Device Architecture Register

[TRCDEVID](#): Device Configuration Register

[TRCDEVID1](#): Device Configuration Register 1

[TRCDEVID2](#): Device Configuration Register 2

[TRCDEVTYPE](#): Device Type Register

[TRCEVENTCTL0R](#): Event Control 0 Register

[TRCEVENTCTL1R](#): Event Control 1 Register

[TRCEXTINSELR<n>](#): External Input Select Register <n>

[TRCIDR0](#): ID Register 0

[TRCIDR1](#): ID Register 1

[TRCIDR10](#): ID Register 10

[TRCIDR11](#): ID Register 11

[TRCIDR12](#): ID Register 12

[TRCIDR13](#): ID Register 13

[TRCIDR2](#): ID Register 2

[TRCIDR3](#): ID Register 3

[TRCIDR4](#): ID Register 4

[TRCIDR5](#): ID Register 5

[TRCIDR6](#): ID Register 6

[TRCIDR7](#): ID Register 7

[TRCIDR8](#): ID Register 8

[TRCIDR9](#): ID Register 9

[TRCIMSPEC0](#): IMP DEF Register 0

[TRCIMSPEC<n>](#): IMP DEF Register <n>

[TRCITCTRL](#): Integration Mode Control Register

[TRCITEEDCR](#): Instrumentation Trace Extension External Debug Control Register

[TRCLAR](#): Lock Access Register

[TRCLSR](#): Lock Status Register

[TRCOSLSR](#): Trace OS Lock Status Register

[TRCPDCR](#): PowerDown Control Register
[TRCPDSR](#): PowerDown Status Register
[TRCPIDR0](#): Peripheral Identification Register 0
[TRCPIDR1](#): Peripheral Identification Register 1
[TRCPIDR2](#): Peripheral Identification Register 2
[TRCPIDR3](#): Peripheral Identification Register 3
[TRCPIDR4](#): Peripheral Identification Register 4
[TRCPIDR5](#): Peripheral Identification Register 5
[TRCPIDR6](#): Peripheral Identification Register 6
[TRCPIDR7](#): Peripheral Identification Register 7
[TRCPRGCTLR](#): Programming Control Register
[TRCQCTLR](#): Q Element Control Register
[TRCRSCTLR<n>](#): Resource Selection Control Register <n>
[TRCRSR](#): Resources Status Register
[TRCSEQEVR<n>](#): Sequencer State Transition Control Register <n>
[TRCSEQRSTEVR](#): Sequencer Reset Control Register
[TRCSEQSTR](#): Sequencer State Register
[TRCSSCCR<n>](#): Single-shot Comparator Control Register <n>
[TRCSSCSR<n>](#): Single-shot Comparator Control Status Register <n>
[TRCSSPCICR<n>](#): Single-shot Processing Element Comparator Input Control Register <n>
[TRCSTALLCTLR](#): Stall Control Register
[TRCSTATR](#): Trace Status Register
[TRCSYNCPR](#): Synchronization Period Register
[TRCTRACEIDR](#): Trace ID Register
[TRCTSCTLR](#): Timestamp Control Register
[TRCVICTLR](#): ViewInst Main Control Register
[TRCVIIECTLR](#): ViewInst Include/Exclude Control Register
[TRCVIPCSSCTLR](#): ViewInst Start/Stop PE Comparator Control Register
[TRCVISSCTLR](#): ViewInst Start/Stop Control Register
[TRCVMIDCCTLR0](#): Virtual Context Identifier Comparator Control Register 0
[TRCVMIDCCTLR1](#): Virtual Context Identifier Comparator Control Register 1
[TRCVMIDCVR<n>](#): Virtual Context Identifier Comparator Value Register <n>

3005/0907/2022 1617:0621

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

External register index by offset

Below are indexes for external registers in the following blocks:

- [AMU](#)
- [CTI](#)
- [Debug](#)
- [ETE](#)
- [GIC CPU interface](#)
- [GIC Distributor](#)
- [GIC ITS control](#)
- [GIC ITS translation](#)
- [GIC Redistributor](#)
- [GIC Virtual CPU interface](#)
- [GIC Virtual interface control](#)
- [MPAM](#)
- [PMU](#)
- [RAS](#)
- [TRBE](#)
- [Timer](#)

In the AMU block:

Offset	Name	Description	Access
$0x000 + (8 * n)$	AMEVCNTR0<n>[31:0]	Activity Monitors Event Counter Registers 0	RO
$0x004 + (8 * n)$	AMEVCNTR0<n>[63:32]	Activity Monitors Event Counter Registers 0	RO
$0x100 + (8 * n)$	AMEVCNTR1<n>[31:0]	Activity Monitors Event Counter Registers 1	RO
$0x104 + (8 * n)$	AMEVCNTR1<n>[63:32]	Activity Monitors Event Counter Registers 1	RO
$0x400 + (4 * n)$	AMEVTYPER0<n>	Activity Monitors Event Type Registers 0	RO
$0x480 + (4 * n)$	AMEVTYPER1<n>	Activity Monitors Event Type Registers 1	RO
0xC00	AMCNTENSET0	Activity Monitors Count Enable Set Register 0	RO
0xC04	AMCNTENSET1	Activity Monitors Count Enable Set Register 1	RO
0xC20	AMCNTENCLR0	Activity Monitors Count Enable Clear Register 0	RO
0xC24	AMCNTENCLR1	Activity Monitors Count Enable Clear Register 1	RO
0xCE0	AMCGCR	Activity Monitors Counter Group Configuration Register	RO
0xE00	AMCFGR	Activity Monitors Configuration Register	RO
0xE04	AMCR	Activity Monitors Control Register	RO
0xE08	AMIIDR	Activity Monitors Implementation Identification Register	RO
0xFA8	AMDEVAFF0	Activity Monitors Device Affinity Register 0	RO
0xFAC	AMDEVAFF1	Activity Monitors Device Affinity Register 1	RO
0xFBC	AMDEVARCH	Activity Monitors Device Architecture Register	RO
0xFCC	AMDEVTYPE	Activity Monitors Device Type Register	RO
0xFD0	AMPIDR4	Activity Monitors Peripheral Identification Register 4	RO
0xFE0	AMPIDR0	Activity Monitors Peripheral Identification Register 0	RO
0xFE4	AMPIDR1	Activity Monitors Peripheral Identification Register 1	RO
0xFE8	AMPIDR2	Activity Monitors Peripheral Identification Register 2	RO

Offset	Name	Description	Access
0xFEC	AMPIDR3	Activity Monitors Peripheral Identification Register 3	RO
0xFF0	AMCIDR0	Activity Monitors Component Identification Register 0	RO
0xFF4	AMCIDR1	Activity Monitors Component Identification Register 1	RO
0xFF8	AMCIDR2	Activity Monitors Component Identification Register 2	RO
0xFFC	AMCIDR3	Activity Monitors Component Identification Register 3	RO

In the CTI block:

Offset	Name	Description	Access
0x000	CTICONTROL	CTI Control register	RW
0x010	CTIINTACK	CTI Output Trigger Acknowledge register	WO
0x014	CTIAPPSET	CTI Application Trigger Set register	RW
0x018	CTIAPPCLEAR	CTI Application Trigger Clear register	WO
0x01C	CTIAPPULSE	CTI Application Pulse register	WO
0x020 + (4 * n)	CTIINEN<n>	CTI Input Trigger to Output Channel Enable registers	RW
0x0A0 + (4 * n)	CTIOUTEN<n>	CTI Input Channel to Output Trigger Enable registers	RW
0x130	CTITRIGINSTATUS	CTI Trigger In Status register	RO
0x134	CTITRIGOUTSTATUS	CTI Trigger Out Status register	RO
0x138	CTICHINSTATUS	CTI Channel In Status register	RO
0x13C	CTICHOUTSTATUS	CTI Channel Out Status register	RO
0x140	CTIGATE	CTI Channel Gate Enable register	RW
0x144	ASICCTL	CTI External Multiplexer Control register	RO
0x150	CTIDEVCTL	CTI Device Control register	RW
0xF00	CTIITCTRL	CTI Integration mode Control register	RW
0xFA0	CTICLAIMSET	CTI CLAIM Tag Set register	RW
0xFA4	CTICLAIMCLR	CTI CLAIM Tag Clear register	RW
0xFA8	CTIDEVAFF0	CTI Device Affinity register 0	RO
0xFAC	CTIDEVAFF1	CTI Device Affinity register 1	RO
0xFB0	CTILAR	CTI Lock Access Register	WO
0xFB4	CTILSR	CTI Lock Status Register	RO
0xFB8	CTIAUTHSTATUS	CTI Authentication Status register	RO
0xFBC	CTIDEVARCH	CTI Device Architecture register	RO
0xFC0	CTIDEVID2	CTI Device ID register 2	RO
0xFC4	CTIDEVID1	CTI Device ID register 1	RO
0xFC8	CTIDEVID	CTI Device ID register 0	RO
0xFCC	CTIDEVTYPE	CTI Device Type register	RO
0xFD0	CTIPIDR4	CTI Peripheral Identification Register 4	RO
0xFE0	CTIPIDR0	CTI Peripheral Identification Register 0	RO
0xFE4	CTIPIDR1	CTI Peripheral Identification Register 1	RO
0xFE8	CTIPIDR2	CTI Peripheral Identification Register 2	RO
0xFEC	CTIPIDR3	CTI Peripheral Identification Register 3	RO
0xFF0	CTICIDR0	CTI Component Identification Register 0	RO
0xFF4	CTICIDR1	CTI Component Identification Register 1	RO
0xFF8	CTICIDR2	CTI Component Identification Register 2	RO

Offset	Name	Description	Access
0xFFC	CTICIDR3	CTI Component Identification Register 3	RO

In the Debug block:

Offset	Name	Description	Access
0x020	EDES	External Debug Event Status Register	RW
0x024	EDEC	External Debug Execution Control Register	RW
0x028	EDSCR2	External Debug Status and Control Register 2	RW
0x030	EDWAR[31:0]	External Debug Watchpoint Address Register	RO
0x034	EDWAR[63:32]	External Debug Watchpoint Address Register	RO
0x03C	EDHSR[63:32]	External Debug Halt Status Register	RO
0x038	EDHSR[31:0]	External Debug HaltingHalt SyndromeStatus Register	RO
0x080	DBGDTRRX_EL0	Debug Data Transfer Register, Receive	RW
0x084	EDITR	External Debug Instruction Transfer Register	WO
0x088	EDSCR	External Debug Status and Control Register	RW
0x08C	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit	RW
0x090	EDRCR	External Debug Reserve Control Register	WO
0x094	EDACR	External Debug Auxiliary Control Register	RW
0x098	EDECCR	External Debug Exception Catch Control Register	RW
0x0A0	EDPCSR[31:0]	External Debug Program Counter Sample Register	RO
0x0A4	EDCIDS	External Debug Context ID Sample Register	RO
0x0A8	EDVIDSR	External Debug Virtual Context Sample Register	RO
0x0AC	EDPCSR[63:32]	External Debug Program Counter Sample Register	RO
0x300	OSLAR_EL1	OS Lock Access Register	WO
0x310	EDPRCR	External Debug Power/Reset Control Register	RW
0x314	EDPRSR	External Debug Processor Status Register	RO
0x400 + (16 * n)	DBGBVR<n>_EL1[63:0]	Debug Breakpoint Value Registers	RW
0x408 + (16 * n)	DBGBCR<n>_EL1	Debug Breakpoint Control Registers	RW
0x800 + (16 * n)	DBGWVR<n>_EL1[63:0]	Debug Watchpoint Value Registers	RW
0x808 + (16 * n)	DBGWCR<n>_EL1	Debug Watchpoint Control Registers	RW
0xD00	MIDR_EL1	Main ID Register	RO
0xD20	EDPFR[31:0]	External Debug Processor Feature Register	RO
0xD24	EDPFR[63:32]	External Debug Processor Feature Register	RO
0xD28	EDDFR[31:0]	External Debug Feature Register	RO
0xD2C	EDDFR[63:32]	External Debug Feature Register	RO
0xD48	EDDFR1	External Debug Feature Register 1	RO
0xD60	EDAA32PFR	External Debug Auxiliary Processor Feature Register	RO
0xF00	EDITCTRL	External Debug Integration mode Control register	RW
0xFA0	DBGCLAIMSET_EL1	Debug CLAIM Tag Set register	RW
0xFA4	DBGCLAIMCLR_EL1	Debug CLAIM Tag Clear register	RW
0xFA8	EDDEVAFF0	External Debug Device Affinity register 0	RO
0xFAC	EDDEVAFF1	External Debug Device Affinity register 1	RO
0xFB0	EDLAR	External Debug Lock Access Register	WO
0xFB4	EDLSR	External Debug Lock Status Register	RO

Offset	Name	Description	Access
0xFB8	DBGAUTHSTATUS_EL1	Debug Authentication Status register	RO
0xFBC	EDDEVARCH	External Debug Device Architecture register	RO
0xFC0	EDDEVID2	External Debug Device ID register 2	RO
0xFC4	EDDEVID1	External Debug Device ID register 1	RO
0xFC8	EDDEVID	External Debug Device ID register 0	RO
0xFCC	EDDEVTYPE	External Debug Device Type register	RO
0xFD0	EDPIDR4	External Debug Peripheral Identification Register 4	RO
0xFE0	EDPIDR0	External Debug Peripheral Identification Register 0	RO
0xFE4	EDPIDR1	External Debug Peripheral Identification Register 1	RO
0xFE8	EDPIDR2	External Debug Peripheral Identification Register 2	RO
0xFEC	EDPIDR3	External Debug Peripheral Identification Register 3	RO
0xFF0	EDCIDR0	External Debug Component Identification Register 0	RO
0xFF4	EDCIDR1	External Debug Component Identification Register 1	RO
0xFF8	EDCIDR2	External Debug Component Identification Register 2	RO
0xFFC	EDCIDR3	External Debug Component Identification Register 3	RO

In the ETE block:

Offset	Name	Description	Access
0x004	TRCPRGCTLR	Programming Control Register	RW
0x00C	TRCSTATR	Trace Status Register	RO
0x010	TRCCONFIGR	Trace Configuration Register	RW
0x018	TRCAUXCTLR	Auxiliary Control Register	RW
0x020	TRCEVENTCTL0R	Event Control 0 Register	RW
0x024	TRCEVENTCTL1R	Event Control 1 Register	RW
0x028	TRCRSR	Resources Status Register	RW
0x02C	TRCSTALLCTLR	Stall Control Register	RW
0x030	TRCTSCTLR	Timestamp Control Register	RW
0x034	TRCSYNCPR	Synchronization Period Register	RW
0x038	TRCCCCTLR	Cycle Count Control Register	RW
0x03C	TRCBBCTLR	Branch Broadcast Control Register	RW
0x040	TRCTRACEIDR	Trace ID Register	RW
0x044	TRCQCTLR	Q Element Control Register	RW
0x048	TRCITEEDCR	Instrumentation Trace Extension External Debug Control Register	RW
0x080	TRCVICTLR	ViewInst Main Control Register	RW
0x084	TRCVIIECTLR	ViewInst Include/Exclude Control Register	RW
0x088	TRCVISSCTLR	ViewInst Start/Stop Control Register	RW
0x08C	TRCVIPCSSCTLR	ViewInst Start/Stop PE Comparator Control Register	RW
0x100 + (4 * n)	TRCSEQEVR<n>	Sequencer State Transition Control Register <n>	RW
0x118	TRCSEQRSTEV	Sequencer Reset Control Register	RW
0x11C	TRCSEQSTR	Sequencer State Register	RW
0x120 + (4 * n)	TRCEXTINSEL<n>	External Input Select Register <n>	RW

Offset	Name	Description	Access
0x140 + (4 * n)	TRCCNTRLDVR<n>	Counter Reload Value Register <n>	RW
0x150 + (4 * n)	TRCCNTCTLR<n>	Counter Control Register <n>	RW
0x160 + (4 * n)	TRCCNTVR<n>	Counter Value Register <n>	RW
0x180	TRCIDR8	ID Register 8	RO
0x184	TRCIDR9	ID Register 9	RO
0x188	TRCIDR10	ID Register 10	RO
0x18C	TRCIDR11	ID Register 11	RO
0x190	TRCIDR12	ID Register 12	RO
0x194	TRCIDR13	ID Register 13	RO
0x1C0	TRCIMSPEC0	IMP DEF Register 0	RW
0x1C0 + (4 * n)	TRCIMSPEC<n>	IMP DEF Register <n>	RW
0x1E0	TRCIDR0	ID Register 0	RO
0x1E4	TRCIDR1	ID Register 1	RO
0x1E8	TRCIDR2	ID Register 2	RO
0x1EC	TRCIDR3	ID Register 3	RO
0x1F0	TRCIDR4	ID Register 4	RO
0x1F4	TRCIDR5	ID Register 5	RO
0x1F8	TRCIDR6	ID Register 6	RO
0x1FC	TRCIDR7	ID Register 7	RO
0x200 + (4 * n)	TRCRSCTLR<n>	Resource Selection Control Register <n>	RW
0x280 + (4 * n)	TRCSSCCR<n>	Single-shot Comparator Control Register <n>	RW
0x2A0 + (4 * n)	TRCSSCSR<n>	Single-shot Comparator Control Status Register <n>	RW
0x2C0 + (4 * n)	TRCSSPICR<n>	Single-shot Processing Element Comparator Input Control Register <n>	RW
0x304	TRCOSLSR	Trace OS Lock Status Register	RO
0x310	TRCPDCR	PowerDown Control Register	RW
0x314	TRCPDSR	PowerDown Status Register	RO
0x400 + (8 * n)	TRCACVR<n>	Address Comparator Value Register <n>	RW
0x480 + (8 * n)	TRCACATR<n>	Address Comparator Access Type Register <n>	RW
0x600 + (8 * n)	TRCCIDCVR<n>	Context Identifier Comparator Value Registers <n>	RW
0x640 + (8 * n)	TRCVMIDCVR<n>	Virtual Context Identifier Comparator Value Register <n>	RW
0x680	TRCCIDCCTLR0	Context Identifier Comparator Control Register 0	RW
0x684	TRCCIDCCTLR1	Context Identifier Comparator Control Register 1	RW
0x688	TRCVMIDCCTLR0	Virtual Context Identifier Comparator Control Register 0	RW
0x68C	TRCVMIDCCTLR1	Virtual Context Identifier Comparator Control Register 1	RW
0xF00	TRCITCTRL	Integration Mode Control Register	RW
0xFA0	TRCCLAIMSET	Claim Tag Set Register	RW
0xFA4	TRCCLAIMCLR	Claim Tag Clear Register	RW
0xFA8	TRCDEVAFF	Device Affinity Register	RO
0xFB0	TRCLAR	Lock Access Register	WO
0xFB4	TRCLSR	Lock Status Register	RO
0xFB8	TRCAUTHSTATUS	Authentication Status Register	RO
0xFBC	TRCDEVARCH	Device Architecture Register	RO
0xFC0	TRCDEVID2	Device Configuration Register 2	RO
0xFC4	TRCDEVID1	Device Configuration Register 1	RO
0xFC8	TRCDEVID	Device Configuration Register	RO

Offset	Name	Description	Access
0xFCC	TRCDEVTYPE	Device Type Register	RO
0xFD0	TRCPIDR4	Peripheral Identification Register 4	RO
0xFD4	TRCPIDR5	Peripheral Identification Register 5	RO
0xFD8	TRCPIDR6	Peripheral Identification Register 6	RO
0xFDC	TRCPIDR7	Peripheral Identification Register 7	RO
0xFE0	TRCPIDR0	Peripheral Identification Register 0	RO
0xFE4	TRCPIDR1	Peripheral Identification Register 1	RO
0xFE8	TRCPIDR2	Peripheral Identification Register 2	RO
0xFEC	TRCPIDR3	Peripheral Identification Register 3	RO
0xFF0	TRCCIDR0	Component Identification Register 0	RO
0xFF4	TRCCIDR1	Component Identification Register 1	RO
0xFF8	TRCCIDR2	Component Identification Register 2	RO
0xFFC	TRCCIDR3	Component Identification Register 3	RO

In the GIC CPU interface block:

Offset	Name	Description	Access
0x0000	GICC_CTLR	CPU Interface Control Register	RW
0x0004	GICC_PMR	CPU Interface Priority Mask Register	RW
0x0008	GICC_BPR	CPU Interface Binary Point Register	RW
0x000C	GICC_IAR	CPU Interface Interrupt Acknowledge Register	RO
0x0010	GICC_EOIR	CPU Interface End Of Interrupt Register	WO
0x0014	GICC_RPR	CPU Interface Running Priority Register	RO
0x0018	GICC_HPIR	CPU Interface Highest Priority Pending Interrupt Register	RO
0x001C	GICC_ABPR	CPU Interface Aliased Binary Point Register	RW
0x0020	GICC_AIAR	CPU Interface Aliased Interrupt Acknowledge Register	RO
0x0024	GICC_AEOIR	CPU Interface Aliased End Of Interrupt Register	WO
0x0028	GICC_AHPIR	CPU Interface Aliased Highest Priority Pending Interrupt Register	RO
0x002C	GICC_STATUSR	CPU Interface Status Register	RW
0x002C	GICC_STATUSR	CPU Interface Status Register	RW
0x00D0 + (4 * n)	GICC_APR<n>	CPU Interface Active Priorities Registers	RW
0x00E0 + (4 * n)	GICC_NSAPR<n>	CPU Interface Non-secure Active Priorities Registers	RW
0x00FC	GICC_IIDR	CPU Interface Identification Register	RO
0x1000	GICC_DIR	CPU Interface Deactivate Interrupt Register	WO

In the GIC Distributor block:

In the Dist_base block:

Offset	Name	Description	Access
0x0000	GICD_CTLR	Distributor Control Register	RW
0x0004	GICD_TYPER	Interrupt Controller Type Register	RO
0x0008	GICD_IIDR	Distributor Implementer Identification Register	RO
0x000C	GICD_TYPER2	Interrupt Controller Type Register 2	RO
0x0010	GICD_STATUSR	Error Reporting Status Register	RW
0x0010	GICD_STATUSR	Error Reporting Status Register	RW

Offset	Name	Description	Access
0x0040	GICD_SETSPI_NSR	Set Non-secure SPI Pending Register	WO
0x0048	GICD_CLRSPI_NSR	Clear Non-secure SPI Pending Register	WO
0x0050	GICD_SETSPI_SR	Set Secure SPI Pending Register	WO
0x0058	GICD_CLRSPI_SR	Clear Secure SPI Pending Register	WO
0x0080 + (4 * n)	GICD_IGROUPR<n>	Interrupt Group Registers	RW
0x0100 + (4 * n)	GICD_ISENABLER<n>	Interrupt Set-Enable Registers	RW
0x0180 + (4 * n)	GICD_ICENABLER<n>	Interrupt Clear-Enable Registers	RW
0x0200 + (4 * n)	GICD_ISPENDR<n>	Interrupt Set-Pending Registers	RW
0x0280 + (4 * n)	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers	RW
0x0300 + (4 * n)	GICD_ISACTIVER<n>	Interrupt Set-Active Registers	RW
0x0380 + (4 * n)	GICD_ICACTIVER<n>	Interrupt Clear-Active Registers	RW
0x0400 + (4 * n)	GICD_IPRIORITYR<n>	Interrupt Priority Registers	RW
0x0800 + (4 * n)	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers	RW
0x0C00 + (4 * n)	GICD_ICFGR<n>	Interrupt Configuration Registers	RW
0x0D00 + (4 * n)	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers	RW
0x0E00 + (4 * n)	GICD_NSACR<n>	Non-secure Access Control Registers	RW
0x0F00	GICD_SGIR	Software Generated Interrupt Register	WO
0x0F10 + (4 * n)	GICD_CPENDSGIR<n>	SGI Clear-Pending Registers	RW
0x0F20 + (4 * n)	GICD_SPENDSGIR<n>	SGI Set-Pending Registers	RW
0x0F80 + (4 * n)	GICD_INMIR<n>	Non-maskable Interrupt Registers, x = 0 to 31	RW
0x1000 + (4 * n)	GICD_IGROUPR<n>E	Interrupt Group Registers (extended SPI range)	RW
0x1200 + (4 * n)	GICD_ISENABLER<n>E	Interrupt Set-Enable Registers	RW
0x1400 + (4 * n)	GICD_ICENABLER<n>E	Interrupt Clear-Enable Registers	RW
0x1600 + (4 * n)	GICD_ISPENDR<n>E	Interrupt Set-Pending Registers (extended SPI range)	RW
0x1800 + (4 * n)	GICD_ICPENDR<n>E	Interrupt Clear-Pending Registers (extended SPI range)	RW
0x1A00 + (4 * n)	GICD_ISACTIVER<n>E	Interrupt Set-Active Registers (extended SPI range)	RW
0x1C00 + (4 * n)	GICD_ICACTIVER<n>E	Interrupt Clear-Active Registers (extended SPI range)	RW
0x2000 + (4 * n)	GICD_IPRIORITYR<n>E	Holds the priority of the corresponding interrupt for each extended SPI supported by the GIC.	RW
0x3000 + (4 * n)	GICD_ICFGR<n>E	Interrupt Configuration Registers (Extended SPI Range)	RW
0x3400 + (4 * n)	GICD_IGRPMODR<n>E	Interrupt Group Modifier Registers (extended SPI range)	RW
0x3600 + (4 * n)	GICD_NSACR<n>E	Non-secure Access Control Registers	RW
0x3B00 + (4 * n)	GICD_INMIR<n>E	Non-maskable Interrupt Registers for Extended SPIs, x = 0 to 31	RW
0x6000 + (8 * n)	GICD_IROUTER<n>	Interrupt Routing Registers	RW
0x8000 + (8 * n)	GICD_IROUTER<n>E	Interrupt Routing Registers (Extended SPI Range)	RW

In the MSI_base block:

Offset	Name	Description	Access
0x0004	GICM_TYPER	Distributor MSI Type Register	RO
0x0040	GICM_SETSPI_NSR	Set Non-secure SPI Pending Register	WO
0x0048	GICM_CLRSPI_NSR	Clear Non-secure SPI Pending Register	WO

Offset	Name	Description	Access
0x0050	GICM_SETSPI_SR	Set Secure SPI Pending Register	WO
0x0058	GICM_CLRSPI_SR	Clear Secure SPI Pending Register	WO
0x0FCC	GICM_IIDR	Distributor Implementer Identification Register	RO

In the GIC ITS control block:

Offset	Name	Description	Access
0x0000	GITS_CTLR	ITS Control Register	RW
0x0004	GITS_IIDR	ITS Identification Register	RO
0x0008	GITS_TYPER	ITS Type Register	RO
0x0010	GITS_MPAMIDR	Report maximum PARTID and PMG Register	RO
0x0014	GITS_PARTIDR	Set PARTID and PMG Register	RW
0x0018	GITS_MPIDR	Report ITS's affinity.	RO
0x0040	GITS_STATUSR	ITS Error Reporting Status Register	RW
0x0048	GITS_UMSIR	ITS Unmapped MSI register	RO
0x0080	GITS_CBASER	ITS Command Queue Descriptor	RW
0x0088	GITS_CWRITER	ITS Write Register	RW
0x0090	GITS_CREADR	ITS Read Register	RO
0x0100 + (8 * n)	GITS_BASER<n>	ITS Translation Table Descriptors	RW
0x20020	GITS_SGIR	ITS SGI Register	WO

In the GIC ITS translation block:

Offset	Name	Description	Access
0x0040	GITS_TRANSLATER	ITS Translation Register	WO

In the GIC Redistributor block:

In the RD_base block:

Offset	Name	Description	Access
0x0000	GICR_CTLR	Redistributor Control Register	RW
0x0004	GICR_IIDR	Redistributor Implementer Identification Register	RO
0x0008	GICR_TYPER	Redistributor Type Register	RO
0x0010	GICR_STATUSR	Error Reporting Status Register	RW
0x0010	GICR_STATUSR	Error Reporting Status Register	RW
0x0014	GICR_WAKER	Redistributor Wake Register	RW
0x0018	GICR_MPAMIDR	Report maximum PARTID and PMG Register	RO
0x001C	GICR_PARTIDR	Set PARTID and PMG Register	RW
0x0040	GICR_SETLPIR	Set LPI Pending Register	WO
0x0048	GICR_CLRLPIR	Clear LPI Pending Register	WO
0x0070	GICR_PROPBASER	Redistributor Properties Base Address Register	RW
0x0078	GICR_PENDBASER	Redistributor LPI Pending Table Base Address Register	RW
0x00A0	GICR_INVLPIR	Redistributor Invalidate LPI Register	WO
0x00B0	GICR_INVALLR	Redistributor Invalidate All Register	WO
0x00C0	GICR_SYNCR	Redistributor Synchronize Register	RO

In the SGI_base block:

Offset	Name	Description	Access
0x0080	GICR_IGROUPR0	Interrupt Group Register 0	RW
0x0080 + (4 * n)	GICR_IGROUPR<n>E	Interrupt Group Registers	RW
0x0100	GICR_ISENABLER0	Interrupt Set-Enable Register 0	RW
0x0100 + (4 * n)	GICR_ISENABLER<n>E	Interrupt Set-Enable Registers	RW
0x0180	GICR_ICENABLER0	Interrupt Clear-Enable Register 0	RW
0x0180 + (4 * n)	GICR_ICENABLER<n>E	Interrupt Clear-Enable Registers	RW
0x0200	GICR_ISPENDR0	Interrupt Set-Pending Register 0	RW
0x0200 + (4 * n)	GICR_ISPENDR<n>E	Interrupt Set-Pending Registers	RW
0x0280	GICR_ICPENDR0	Interrupt Clear-Pending Register 0	RW
0x0280 + (4 * n)	GICR_ICPENDR<n>E	Interrupt Clear-Pending Registers	RW
0x0300	GICR_ISACTIVER0	Interrupt Set-Active Register 0	RW
0x0300 + (4 * n)	GICR_ISACTIVER<n>E	Interrupt Set-Active Registers	RW
0x0380	GICR_ICACTIVER0	Interrupt Clear-Active Register 0	RW
0x0380 + (4 * n)	GICR_ICACTIVER<n>E	Interrupt Clear-Active Registers	RW
0x0400 + (4 * n)	GICR_IPRIORITYR<n>	Interrupt Priority Registers	RW
0x0400 + (4 * n)	GICR_IPRIORITYR<n>E	Interrupt Priority Registers (extended PPI range)	RW
0x0400 + (4 * n)	GICR_IPRIORITYR<n>	Interrupt Priority Registers	RW
0x0C00	GICR_ICFGR0	Interrupt Configuration Register 0	RW
0x0C00 + (4 * n)	GICR_ICFGR<n>E	Interrupt configuration registers	RW
0x0C04	GICR_ICFGR1	Interrupt Configuration Register 1	RW
0x0D00	GICR_IGRPMODR0	Interrupt Group Modifier Register 0	RW
0x0D00 + (4 * n)	GICR_IGRPMODR<n>E	Interrupt Group Modifier Registers	RW
0x0E00	GICR_NSACR	Non-secure Access Control Register	RW
0x0F80	GICR_INMIRO	Non-maskable Interrupt Register for PPIs.	RW
0x0F80 + (4 * n)	GICR_INMIR<n>E	Non-maskable Interrupt Registers for Extended PPIs, x = 1 to 2.	RW

In the VLPI_base block:

Offset	Name	Description	Access
0x0070	GICR_VPROPBASER	Virtual Redistributor Properties Base Address Register	RW
0x0078	GICR_VPENDBASER	Virtual Redistributor LPI Pending Table Base Address Register	RW
0x0080	GICR_VSGIR	Redistributor virtual SGI pending state request register	WO
0x0088	GICR_VSGIPENDR	Redistributor virtual SGI pending state register	RO

In the GIC Virtual CPU interface block:

Offset	Name	Description	Access
0x0000	GICV_CTLR	Virtual Machine Control Register	RW
0x0004	GICV_PMR	Virtual Machine Priority Mask Register	RW
0x0008	GICV_BPR	Virtual Machine Binary Point Register	RW
0x000C	GICV_IAR	Virtual Machine Interrupt Acknowledge Register	RO
0x0010	GICV_EOIR	Virtual Machine End Of Interrupt Register	WO
0x0014	GICV_RPR	Virtual Machine Running Priority Register	RO
0x0018	GICV_HPPIR	Virtual Machine Highest Priority Pending Interrupt Register	RO
0x001C	GICV_ABPR	Virtual Machine Aliased Binary Point Register	RW

Offset	Name	Description	Access
0x0020	GICV_AIAR	Virtual Machine Aliased Interrupt Acknowledge Register	RO
0x0024	GICV_AEOIR	Virtual Machine Aliased End Of Interrupt Register	WO
0x0028	GICV_AHPPIR	Virtual Machine Aliased Highest Priority Pending Interrupt Register	RO
0x002C	GICV_STATUSR	Virtual Machine Error Reporting Status Register	RW
0x00D0 + (4 * n)	GICV_APR<n>	Virtual Machine Active Priorities Registers	RW
0x00FC	GICV_IIDR	Virtual Machine CPU Interface Identification Register	RO
0x1000	GICV_DIR	Virtual Machine Deactivate Interrupt Register	WO

In the GIC Virtual interface control block:

Offset	Name	Description	Access
0x0000	GICH_HCR	Hypervisor Control Register	RW
0x0004	GICH_VTR	Virtual Type Register	RO
0x0008	GICH_VMCR	Virtual Machine Control Register	RW
0x0010	GICH_MISR	Maintenance Interrupt Status Register	RO
0x0020	GICH_EISR	End Interrupt Status Register	RO
0x0030	GICH_ELRSR	Empty List Register Status Register	RO
0x00F0 + (4 * n)	GICH_APR<n>	Active Priorities Registers	RW
0x0100 + (4 * n)	GICH_LR<n>	List Registers	RW

In the MPAM block:

In the MPAMF_BASE_ns block:

Offset	Name	Description	Access
0x0000	MPAMF_IDR	MPAM Features Identification Register	RO
0x0018	MPAMF_IIDR	MPAM Implementation Identification Register	RO
0x0020	MPAMF_AIDR	MPAM Architecture Identification Register	RO
0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register	RO
0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register	RO
0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register	RO
0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register	RO
0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register	RO
0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register	RO
0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register	RO
0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register	RO
0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register	RO
0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register	RW

Offset	Name	Description	Access
0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register	RW
0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register	RW
0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register	RW
0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register	RW
0x00F0	MPAMF_ECR	MPAM Error Control Register	RW
0x00F8	MPAMF_ESR	MPAM Error Status Register	RW
0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register	RW
0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register	RW
0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register	RW
0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register	RW
0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register	RW
0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register	RW
0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register	RW
0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register	WO/ RAZ
0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register	WO/ RAZ
0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register	RW
0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register	RW
0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register	RW
0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register	RW
0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register	RW
0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register	WO/ RAZ
0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register	RW
0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register	RW
0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register	RW
0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register	RW
0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register	RW
0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register	RW
0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register	RO

Offset	Name	Description	Access
0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register	RW
0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register	RW
0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register	RW
0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register	RW
0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register	RO
0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register	RW
0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register	RW
0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register	RW
0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register	RW
0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register	RW
0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register	RO
0x1000 + (4 * n)	MPAMCFG_CPBM<n>	MPAM Cache Portion Bitmap Partition Configuration Register	RW
0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register	RW

In the MPAMF_BASE_r1 block:

Offset	Name	Description	Access
0x0000	MPAMF_IDR	MPAM Features Identification Register	RO
0x0018	MPAMF_IIDR	MPAM Implementation Identification Register	RO
0x0020	MPAMF_AIDR	MPAM Architecture Identification Register	RO
0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register	RO
0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register	RO
0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register	RO
0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register	RO
0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register	RO
0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register	RO
0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register	RO
0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register	RO
0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register	RO
0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register	RW
0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register	RW

Offset	Name	Description	Access
0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register	RW
0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register	RW
0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register	RW
0x00F0	MPAMF_ECR	MPAM Error Control Register	RW
0x00F8	MPAMF_ESR	MPAM Error Status Register	RW
0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register	RW
0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register	RW
0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register	RW
0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register	RW
0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register	RW
0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register	RW
0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register	RW
0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register	WO/ RAZ
0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register	WO/ RAZ
0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register	RW
0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register	RW
0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register	RW
0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register	RW
0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register	RW
0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register	WO/ RAZ
0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register	RW
0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register	RW
0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register	RW
0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register	RW
0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register	RW
0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register	RW
0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register	RO
0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register	RW

Offset	Name	Description	Access
0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register	RW
0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register	RW
0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register	RW
0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register	RO
0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register	RW
0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register	RW
0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register	RW
0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register	RW
0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register	RW
0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register	RO
0x1000 + (4 * n)	MPAMCFG_CPBM<n>	MPAM Cache Portion Bitmap Partition Configuration Register	RW
0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register	RW

In the MPAMF_BASE_rt block:

Offset	Name	Description	Access
0x0000	MPAMF_IDR	MPAM Features Identification Register	RO
0x0018	MPAMF_IIDR	MPAM Implementation Identification Register	RO
0x0020	MPAMF_AIDR	MPAM Architecture Identification Register	RO
0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register	RO
0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register	RO
0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register	RO
0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register	RO
0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register	RO
0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register	RO
0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register	RO
0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register	RO
0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register	RO
0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register	RW
0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register	RW
0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register	RW

Offset	Name	Description	Access
0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register	RW
0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register	RW
0x00F0	MPAMF_ECR	MPAM Error Control Register	RW
0x00F8	MPAMF_ESR	MPAM Error Status Register	RW
0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register	RW
0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register	RW
0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register	RW
0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register	RW
0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register	RW
0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register	RW
0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register	RW
0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register	WO/ RAZ
0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register	WO/ RAZ
0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register	RW
0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register	RW
0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register	RW
0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register	RW
0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register	RW
0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register	WO/ RAZ
0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register	RW
0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register	RW
0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register	RW
0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register	RW
0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register	RW
0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register	RW
0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register	RO
0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register	RW
0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register	RW

Offset	Name	Description	Access
0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register	RW
0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register	RW
0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register	RO
0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register	RW
0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register	RW
0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register	RW
0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register	RW
0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register	RW
0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register	RO
0x1000 + (4 * n)	MPAMCFG_CPBM<n>	MPAM Cache Portion Bitmap Partition Configuration Register	RW
0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register	RW

In the MPAMF_BASE_s block:

Offset	Name	Description	Access
0x0000	MPAMF_IDR	MPAM Features Identification Register	RO
0x0008	MPAMF_SIDR	MPAM Features Secure Identification Register	RO
0x0018	MPAMF_IIDR	MPAM Implementation Identification Register	RO
0x0020	MPAMF_AIDR	MPAM Architecture Identification Register	RO
0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register	RO
0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register	RO
0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register	RO
0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register	RO
0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register	RO
0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register	RO
0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register	RO
0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register	RO
0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register	RO
0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register	RW
0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register	RW
0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register	RW

Offset	Name	Description	Access
0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register	RW
0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register	RW
0x00F0	MPAMF_ECR	MPAM Error Control Register	RW
0x00F8	MPAMF_ESR	MPAM Error Status Register	RW
0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register	RW
0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register	RW
0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register	RW
0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register	RW
0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register	RW
0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register	RW
0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register	RW
0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register	WO/ RAZ
0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register	WO/ RAZ
0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register	RW
0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register	RW
0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register	RW
0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register	RW
0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register	RW
0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register	WO/ RAZ
0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register	RW
0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register	RW
0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register	RW
0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register	RW
0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register	RW
0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register	RW
0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register	RO
0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register	RW
0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register	RW

Offset	Name	Description	Access
0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register	RW
0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register	RW
0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register	RO
0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register	RW
0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register	RW
0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register	RW
0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register	RW
0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register	RW
0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register	RO
0x1000 + (4 * n)	MPAMCFG_CPBM<n>	MPAM Cache Portion Bitmap Partition Configuration Register	RW
0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register	RW

In the PMU block:

Offset	Name	
0x000 + (8 * n) for n in 30:0	PMEVCNTR<n>_EL0	P
0x0F8	PMCCNTR_EL0 PMCCNTR_EL0[31:0]	P
0x100 0x0FC	PMICNTR_EL0 PMCCNTR_EL0[63:32]	P I
0x200	PMPCSR PMPCSR[31:0]	P
0x208 0x204	PMCID1SR PMPCSR[63:32]	C S
0x208	PMVCIDSR PMCID1SR	C I
0x20C	PMVIDSR	V
0x220	PMPCSR PMPCSR[31:0]	P

Offset	Name
0x2280x224	PMCID1SRPMPCSR[63:32]
0x228	PMCCIDSRPMCID1SR
0x22C	PMCID2SR
0x2300x400 + (4 * n)	PMPCSCTLPMEVTYPEPER<n>_EL0[31:0]
0x400 + (4 * n) for n in 30:00x47C	PMEVTYPEPER<n>_EL0[31:0]PMCCFILTR_EL0
0x4000xA00 + (84 * n) for n in 30:0	PMEVTYPEPER<n>_EL0[63:0]PMEVTYPEPER<n>_EL0[63:32]
0x47C0xC00	PMCCFILTR_EL0PMCNTENSET_EL0
0x4800xC20	PMICFILTR_EL0PMCNTENCLR_EL0
0x4F80xC40	PMCCFILTR_EL0PMINTENSET_EL1
0x600 + (8 * n) for n in 30:00x600	PMEVCNTSVR<n>_EL1PMINTENCLR_EL1
0x6F80xC80	PMCCNTSVR_EL1PMOVSCCLR_EL0
0x7000xCA0	PMICNTSVR_EL1PMSWINC_EL0
0xA00 + (4 * n) for n in 30:00xCC0	PMEVTYPEPER<n>_EL0[63:32]PMOVSSSET_EL0
0xC000xE00	PMCNTENSET_EL0PMCFGR
0xC040xE04	PMCNTENSET_EL0[63:32]PMCR_EL0
0xC100xE20	PMCNTENPMCEID0
0xC200xE24	PMCNTENCLR_EL0PMCEID1
0xC240xE28	PMCNTENCLR_EL0[63:32]PMCEID2

Offset	Name
0xC400xE2C	PMINTENSET_EL1PMCEID3
0xC440xE40	PMINTENSET_EL1[63:32]PMMIR
0xC500xF00	PMINTENPMITCTRL
0xC600xFA8	PMINTENCLR_EL1PMDEVAFF0
0xC640xFAC	PMINTENCLR_EL1[63:32]PMDEVAFF1
0xC800xFB0	PMOVSLR_EL0PMLAR
0xC840xFB4	PMOVSLR_EL0[63:32]PMLSR
0xC900xFB8	PMOVS PMAUTHSTATUS
0xCA00xFBC	PMSWINC_EL0PMDEVARCH
0xCC00xFC8	PMOVSSET_EL0PMDEVID
0xCC40xFCC	PMOVSSET_EL0[63:32]PMDEVTYPE
0xCE00xFD0	PMCGCR0PMPIDR4
0xE000xFE0	PMCFGRPMPIDR0
0xE040xFE4	PMCR_EL0PMPIDR1
0xE080xFE8	PMIIDR PMPIDR2
0xE100xFEC	PMCR_EL0PMPIDR3
0xE200xFF0	PMCEID0PMCIDR0

Offset	Name	
0xE240xFF4	PMCEID1PMCIDR1	I C I
0xE280xFF8	PMCEID2PMCIDR2	I C I
0xE2C0xFFC	PMCEID3PMCIDR3	I C I
0xE30	PMSSCR_EL1	I S
0xE40	PMMIR	I I
0xF00	PMITCTRL	I r
0xFA8	PMDEVAFF	I r
0xFA8	PMDEVAFF0	I r
0xFAC	PMDEVAFF1	I r
0xFB0	PMLAR	I I
0xFB4	PMLSR	I I
0xFB8	PMAUTHSTATUS	I S
0xFBC	PMDEVARCH	I A
0xFC8	PMDEVID	I r
0xFCC	PMDEVTYPE	I r
0xFD0	PMPIDR4	I I
0xFE0	PMPIDR0	I I

Offset	Name	
0xFE4	PMPIDR1	
0xFE8	PMPIDR2	
0xFEC	PMPIDR3	
0xFF0	PMCIDR0	
0xFF4	PMCIDR1	
0xFF8	PMCIDR2	
0xFFC	PMCIDR3	

In the RAS block:

Offset	Name	Description	Access
0x000 + (64 * n)	ERR<n>FR	Error Record <n> Feature Register	RO
0x008 + (64 * n)	ERR<n>CTLR	Error Record <n> Control Register	RW
0x010 + (64 * n)	ERR<n>STATUS	Error Record <n> Primary Status Register	RW
0x018 + (64 * n)	ERR<n>ADDR	Error Record <n> Address Register	RW
0x020 + (64 * n)	ERR<n>MISC0	Error Record <n> Miscellaneous Register 0	RW
0x028 + (64 * n)	ERR<n>MISC1	Error Record <n> Miscellaneous Register 1	RW
0x030 + (64 * n)	ERR<n>MISC2	Error Record <n> Miscellaneous Register 2	RW
0x038 + (64 * n)	ERR<n>MISC3	Error Record <n> Miscellaneous Register 3	RW
0x800 + (64 * n)	ERR<n>PFGF	Error Record <n> Pseudo-fault Generation Feature Register	RO
0x800 + (8 * n)	ERRIMPDEF<n>	IMPLEMENTATION DEFINED Register <n>	RW
0x808 + (64 * n)	ERR<n>PFGCTL	Error Record <n> Pseudo-fault Generation Control Register	RW
0x810 + (64 * n)	ERR<n>PFGCDN	Error Record <n> Pseudo-fault Generation Countdown Register	RW

Offset	Name	Description	Access
0xE00	ERRGSR	Error Group Status Register	RO
0xE10	ERRIIDR	Implementation Identification Register	RO
0xE80	ERRFHICR0	Fault Handling Interrupt Configuration Register 0	RW
0xE80 + (8 * n)	ERRIRQCR<n>	Generic Error Interrupt Configuration Register <n>	RW
0xE88	ERRFHICR1	Fault Handling Interrupt Configuration Register 1	RW
0xE8C	ERRFHICR2	Fault Handling Interrupt Configuration Register 2	RW
0xE90	ERRERICR0	Error Recovery Interrupt Configuration Register 0	RW
0xE98	ERRERICR1	Error Recovery Interrupt Configuration Register 1	RW
0xE9C	ERRERICR2	Error Recovery Interrupt Configuration Register 2	RW
0xEA0	ERRCRICR0	Critical Error Interrupt Configuration Register 0	RW
0xEA8	ERRCRICR1	Critical Error Interrupt Configuration Register 1	RW
0xEAC	ERRCRICR2	Critical Error Interrupt Configuration Register 2	RW
0xEF8	ERRIRQSR	Error Interrupt Status Register	RW
0xFA8	ERRDEVAFF	Device Affinity Register	RO
0xFBC	ERRDEVARCH	Device Architecture Register	RO
0xFC8	ERRDEVID	Device Configuration Register	RO
0xFD0	ERRPIDR4	Peripheral Identification Register 4	RO
0xFE0	ERRPIDR0	Peripheral Identification Register 0	RO
0xFE4	ERRPIDR1	Peripheral Identification Register 1	RO
0xFE8	ERRPIDR2	Peripheral Identification Register 2	RO
0xFEC	ERRPIDR3	Peripheral Identification Register 3	RO
0xFF0	ERRCIDR0	Component Identification Register 0	RO
0xFF4	ERRCIDR1	Component Identification Register 1	RO
0xFF8	ERRCIDR2	Component Identification Register 2	RO
0xFFC	ERRCIDR3	Component Identification Register 3	RO

In the TRBE block:

Offset	Name	Description	Access
0x000	TRBBASER_EL1	Trace Buffer Base Address Register	RW
0x008	TRBPTR_EL1	Trace Buffer Write Pointer Register	RW
0x010	TRBLIMITR_EL1	Trace Buffer Limit Address Register	RW
0x018	TRBSR_EL1	Trace Buffer Status/syndrome Register	RW
0x020	TRBTRG_EL1	Trace Buffer Trigger Counter Register	RW
0x028	TRBMAR_EL1	Trace Buffer Memory Attribute Register	RW
0x030	TRBIDR_EL1	Trace Buffer ID Register	RO
0x038	TRBCR	Trace Buffer Control Register	RW
0x040	TRBMPAM	Trace Buffer MPAM Configuration Register	RW
0xF00	TRBITCTRL	Integration Mode Control Register	RW
0xFA8	TRBDEVAFF	Device Affinity Register	RO
0xFB0	TRBLAR	Lock Access Register	WO
0xFB4	TRBLSR	Lock Status Register	RO
0xFB8	TRBAUTHSTATUS	Authentication Status Register	RO
0xFBC	TRBDEVARCH	Trace Buffer Device Architecture Register	RO
0xFC0	TRBDEVID2	Device Configuration Register 2	RO
0xFC4	TRBDEVID1	Device Configuration Register 1	RO
0xFC8	TRBDEVID	Device Configuration Register	RO
0xFCC	TRBDEVTYPE	Device Type Register	RO
0xFD0	TRBPIDR4	Peripheral Identification Register 4	RO

Offset	Name	Description	Access
0xFD4	TRBPIDR5	Peripheral Identification Register 5	RO
0xFD8	TRBPIDR6	Peripheral Identification Register 6	RO
0xFDC	TRBPIDR7	Peripheral Identification Register 7	RO
0xFE0	TRBPIDR0	Peripheral Identification Register 0	RO
0xFE4	TRBPIDR1	Peripheral Identification Register 1	RO
0xFE8	TRBPIDR2	Peripheral Identification Register 2	RO
0xFEC	TRBPIDR3	Peripheral Identification Register 3	RO
0xFF0	TRBCIDR0	Component Identification Register 0	RO
0xFF4	TRBCIDR1	Component Identification Register 1	RO
0xFF8	TRBCIDR2	Component Identification Register 2	RO
0xFFC	TRBCIDR3	Component Identification Register 3	RO

In the Timer block:

In the CNTBaseN block:

Offset	Name	Description	Access
0x000	CNTPCT[31:0]	Counter-timer Physical Count	RO
0x004	CNTPCT[63:32]	Counter-timer Physical Count	RO
0x008	CNTVCT[31:0]	Counter-timer Virtual Count	RO
0x00C	CNTVCT[63:32]	Counter-timer Virtual Count	RO
0x010	CNTFRQ	Counter-timer Frequency	RO
0x014	CNTEL0ACR	Counter-timer EL0 Access Control Register	RW
0x018	CNTVOFF[31:0]	Counter-timer Virtual Offset	RO
0x01C	CNTVOFF[63:32]	Counter-timer Virtual Offset	RO
0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer CompareValue	RW
0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer CompareValue	RW
0x028	CNTP_TVAL	Counter-timer Physical Timer TimerValue	RW
0x02C	CNTP_CTL	Counter-timer Physical Timer Control	RW
0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer CompareValue	RW
0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer CompareValue	RW
0x038	CNTV_TVAL	Counter-timer Virtual Timer TimerValue	RW
0x03C	CNTV_CTL	Counter-timer Virtual Timer Control	RW
0xFD0 + (4 * n)	CounterID<n>	Counter ID registers	RO

In the CNTCTLBase block:

Offset	Name	Description	Access
0x000	CNTFRQ	Counter-timer Frequency	RO
0x004	CNTNSAR	Counter-timer Non-secure Access Register	RW
0x008	CNTTIDR	Counter-timer Timer ID Register	RO
0x040 + (4 * n)	CNTACR<n>	Counter-timer Access Control Registers	RW
0x080 + (8 * n)	CNTVOFF<n>[31:0]	Counter-timer Virtual Offsets	RW
0x084 + (8 * n)	CNTVOFF<n>[63:32]	Counter-timer Virtual Offsets	RW
0xFD0 + (4 * n)	CounterID<n>	Counter ID registers	RO

In the CNTControlBase block:

Offset	Name	Description	Access
0x000	CNTCR	Counter Control Register	RW
0x004	CNTSR	Counter Status Register	RO
0x008	CNTCV[63:0]	Counter Count Value register	RW
0x020	CNTFID0	Counter Frequency ID	ImplementationDefined:RO,RW
0x020 + (4 * n)	CNTFID<n>	Counter Frequency IDs, n > 0	ImplementationDefined:RO,RW
0x10	CNTSCR	Counter Scale Register	RW
0x1C	CNTID	Counter Identification Register	RO
0xFD0 + (4 * n)	CounterID<n>	Counter ID registers	RO

In the CNTELOBaseN block:

Offset	Name	Description	Access
0x000	CNTPCT[31:0]	Counter-timer Physical Count	RO
0x004	CNTPCT[63:32]	Counter-timer Physical Count	RO
0x008	CNTVCT[31:0]	Counter-timer Virtual Count	RO
0x00C	CNTVCT[63:32]	Counter-timer Virtual Count	RO
0x010	CNTFRQ	Counter-timer Frequency	RO
0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer CompareValue	RW
0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer CompareValue	RW
0x028	CNTP_TVAL	Counter-timer Physical Timer TimerValue	RW
0x02C	CNTP_CTL	Counter-timer Physical Timer Control	RW
0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer CompareValue	RW
0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer CompareValue	RW
0x038	CNTV_TVAL	Counter-timer Virtual Timer TimerValue	RW
0x03C	CNTV_CTL	Counter-timer Virtual Timer Control	RW
0xFD0 + (4 * n)	CounterID<n>	Counter ID registers	RO

In the CNTReadBase block:

Offset	Name	Description	Access
0x000	CNTCV[63:0]	Counter Count Value register	RO
0xFD0 + (4 * n)	CounterID<n>	Counter ID registers	RO

3005/0907/2022 1517:5809; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGAUTHSTATUS_EL1, Debug Authentication Status register

The DBGAUTHSTATUS_EL1 characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

Configuration

External register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch64 System register [DBGAUTHSTATUS_EL1\[31:0\]](#).

External register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGAUTHSTATUS\[31:0\]](#).

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

DBGAUTHSTATUS_EL1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0			RTNID		RTID		RES0					RLNID		RLID		RES0			SNID		SID		NSNID		NSID						

Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTID.

RTID, bits [25:24]

Root invasive debug.

RTID	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRootInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRootInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is ~~00~~, 0b00.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLID.

RLID, bits [13:12]

Realm invasive debug.

RLID	Meaning
0b00	Not implemented.
0b10	Implemented and disabled. ExternalRealmInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalRealmInvasiveDebugEnabled() == TRUE.

All other values are reserved.

If FEAT_RME is not implemented, the only permitted value is ~~00~~, 0b00.

Bits [11:8]

Reserved, RES0.

SNID, bits [7:6]**When FEAT_Debugv8p4 is implemented:**

Secure non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.SID.

Otherwise:

Secure non-invasive debug.

SNID	Meaning
0b00	Not implemented. One EL3 is not implemented and the following Effective is value true: of SCR_EL3.NS is 1. <ul style="list-style-type: none"> EL3 is not implemented and the Effective value of SCR_EL3.NS is 1. FEAT_RME is implemented without Secure state.
0b10	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

SID, bits [5:4]

Secure invasive debug.

SID	Meaning
0b00	Not implemented. One of the following Effective values is true: <ul style="list-style-type: none"> EL3 is not implemented and the Effective value of SCR_EL3.NS is 1. FEAT_RME is implemented without Secure state.
0b10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSNID, bits [3:2]

When FEAT_Debugv8p4 is implemented:

Non-secure non-invasive debug.

NSNID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

If the Effective value of SCR_EL3.NS is 1, or if EL3 is implemented and EL2 is not implemented, this field reads as 0b11.

All other values are reserved.

Otherwise:

Non-secure non-invasive debug.

NSNID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

All other values are reserved.

NSID, bits [1:0]

Non-secure invasive debug.

NSID	Meaning
0b00	Not implemented. EL3 is not implemented and the Effective value of SCR_EL3.NS is 0.
0b10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
0b11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

All other values are reserved.

Accessing DBGAUTHSTATUS_EL1

DBGAUTHSTATUS_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

Debug	0xFB8	DBGAUTHSTATUS_EL1
-------	-------	-------------------

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n>_EL1 characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register [DBGBVR<n>_EL1](#).

Configuration

External register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to AArch64 System register [DBGBCR<n>_EL1\[31:0\]](#).

External register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGBCR<n>\[31:0\]](#).

DBGBCR<n>_EL1 is in the Core power domain.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

DBGBCR<n>_EL1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LBN	XRES0	RES0	BT	MASK	LBN	BT	SSC	LBN	HMC	SSC	RES0	HMC	BAS	RES0	BAS	PMC	RES0	BT	PMC	E											

When the E field is zero, all the other fields in the register are ignored.

LBNX, Bits bits [31:3024]

When FEAT_Debugv8p9 is implemented:

Linked Breakpoint Number.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

For all other breakpoint types, this field is ignored and reads of the register return an UNKNOWN value.

This field extends DBGBCR<n>_EL1.LBN to support up to 64 implemented breakpoints.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [29]

Reserved, RES0.

MASK, bits [28:24]

When FEAT_ABLE is implemented:

Address Mask. Only address ranges up to 2GB can be watched using a single mask.

MASK	Meaning
0b00000	No mask.
0b00001	Reserved.
0b00010	Reserved.
0b00011..0b11111	Number address bits masked.

Indicates the number of masked address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type.

BT	Meaning	Applies when
0b0000	Unlinked instruction address match. DBGBVR<n>_EL1 is the address of an instruction.	
0b0001	Linked instruction address match. As 0b0000, but linked to a breakpointContext thatmatching has linking enabled.breakpoint.	
0b0010	Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, the Effective value of HCR_EL2.E2H is 1, and either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL1 value. Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of HCR_EL2.E2H is 1, if either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL1 value.	When breakpoint n is context-aware
0b0011	As 0b0010, with linking enabled.	When breakpoint n is context-aware
0b0100	Unlinked instruction address mismatch. DBGBVR<n>_EL1 is the address of an instruction.instruction to be stepped.	When FEAT_ABLE is implemented or EL1 is using AArch32
0b0101	Linked instruction address mismatch. As 0b0100, but linked to a breakpointContext thatmatching has linking enabled.breakpoint.	When FEAT_ABLE is implemented or EL1 is using AArch32
0b0110	Unlinked CONTEXTIDR_EL1 match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1 .	When FEAT_VHE is implemented and breakpoint n is context-aware
0b0111	As 0b0110, with linking enabled.	When FEAT_VHE is implemented and breakpoint n is context-aware
0b1000	Unlinked VMID match. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID .	When EL2 is implemented and breakpoint n is context-aware

0b1001	As 0b1000, with linking enabled.	When EL2 is implemented and breakpoint n is context-aware
0b1010	Unlinked VMID and Context ID match. DBGBCR<n>_EL1 .ContextID is a Context ID compared against CONTEXTIDR_EL1 , and DBGBCR<n>_EL1 .VMID is a VMID compared against VTTBR_EL2 .VMID.	When EL2 is implemented and breakpoint n is context-aware
0b1011	As 0b1010, with linking enabled.	When EL2 is implemented and breakpoint n is context-aware
0b1100	Unlinked CONTEXTIDR_EL2 match. DBGBCR<n>_EL1 .ContextID2 is a Context ID compared against CONTEXTIDR_EL2 .	When FEAT_VHE is implemented and breakpoint n is context-aware
0b1101	As 0b1100, with linking enabled.	When FEAT_VHE is implemented and breakpoint n is context-aware
0b1110	Unlinked Full Context ID match. DBGBCR<n>_EL1 .ContextID is compared against CONTEXTIDR_EL1 , and DBGBCR<n>_EL1 .ContextID2 is compared against CONTEXTIDR_EL2 .	When FEAT_VHE is implemented and breakpoint n is context-aware
0b1111	As 0b1110, with linking enabled.	When FEAT_VHE is implemented and breakpoint n is context-aware

Constraints on breakpoint programming mean some values are reserved under certain conditions.

For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' and 'Reserved DBGBCR<n>_EL1.BT values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked Breakpoint Number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value.

For this all other breakpoint types, this field is ignored and when read the value of the DBGBCR<n>_EL1 register is return an UNKNOWN value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values'.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see [DBGBCR<n>_EL1.SSC](#) description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

When AArch32 is supported:

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints in either AArch32 or AArch64 state, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0b0011	DBGBVR<n>_EL1	Use for T32 instructions
0b1100	DBGBVR<n>_EL1 + 2	Use for T32 instructions
0b1111	DBGBVR<n>_EL1	Use for A64 and A32 instructions

All other values are reserved.

For more information, see 'Using the BAS field in Address Match breakpoints'.

For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0b0000	-	Use for a match anywhere breakpoint
0b0011	DBGBVR<n>_EL1	Use for stepping T32 instructions
0b1100	DBGBVR<n>_EL1	Use for stepping T32 instructions
	+ 2	
0b1111	DBGBVR<n>_EL1	Use for stepping A64 and A32 instructions

For more information, see 'Using the BAS field in Address Match breakpoints'.

For Context matching breakpoints, this field is RES1 and ignored.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

Bit [4:3]

Reserved, RES0.

BT2, bit [3]

When FEAT_ABLE is implemented:

Breakpoint Type 2. With DBGBCR<n>_EL1.BT, specifies breakpoint type.

BT2	Meaning
0b0	As DBGBCR<n>_EL1.BT.
0b1	As DBGBCR<n>_EL1.BT, but with linking enabled.
	This value is only defined for the following DBGBCR<n>_EL1.BT values: 0b0000, 0b0001, 0b0100, and 0b0101.
	All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the [DBGBCR<n>_EL1.SSC](#) description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

(old)

htmldiff from-

(new)

DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n>_EL1 characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register [DBGWVR<n>_EL1](#).

Configuration

External register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to AArch64 System register [DBGWCR<n>_EL1\[31:0\]](#).

External register DBGWCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGWCR<n>\[31:0\]](#).

DBGWCR<n>_EL1 is in the Core power domain.

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

DBGWCR<n>_EL1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LBN	XRES0	RES0	MASK	MASK	RES0	RES0	WT	WT	LBN	LBN	SSC	SSC	CHM	CHM	CBAS																

When the E field is zero, all the other fields in the register are ignored.

LBNX, Bits bits [31:3029]

When FEAT_Debugv8p9 is implemented:

Linked Breakpoint Number.

For Linked data address watchpoints, with DBGWCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.

This field extends DBGWCR<n>_EL1.LBN to support up to 64 implemented breakpoints.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [29]

Reserved, RES0.

MASK, bits [28:24]

Address ~~Mask~~.mask. Only objects up to 2GB can be watched using a single mask.

MASK	Meaning
0b00000	No mask.
0b00001	Reserved.
0b00010	Reserved.

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0b0	Unlinked data address match.
0b1	Linked data address match.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LBN, bits [19:16]

Linked Breakpointbreakpoint Number.number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

For Linked data address watchpoints, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by [DBGWVR<n>_EL1](#) is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>_EL1
xxxxxx1x	Match byte at DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at DBGWVR<n>_EL1 + 2
xxx1xxx	Match byte at DBGWVR<n>_EL1 + 3

In cases where [DBGWVR<n>_EL1](#) addresses a double-word:

BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at DBGWVR<n>_EL1 + 7

If [DBGWVR<n>_EL1\[2\]](#) == 1, only BAS[3:0] is used. Arm deprecates setting [DBGWVR<n>_EL1\[2\]](#) == 1.

The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>.BAS values'.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
0b01	Match instructions that load from a watchpointed address.
0b10	Match instructions that store to a watchpointed address.
0b11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

(old)

htmldiff from-

(new)

EDDEVARCH, External Debug Device Architecture register

The EDDEVARCH characteristics are:

Purpose

Identifies the programmers' model architecture of the external debug component.

Configuration

Implementation of this register is OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

EDDEVARCH is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											

ARCHITECT, bits [31:21]

Defines the architecture of the component. For debug, this is Arm Limited.

Bits [31:28] are the JEP106 continuation code, 0x4.

Bits [27:21] are the JEP106 ID code, 0x3B.

Reads as 0b01000111011.

Access to this field is **RO**.

PRESENT, bit [20]

Indicates that the DEVARCH is present.

Reads as 0b1.

Access to this field is **RO**.

REVISION, bits [19:16]

Defines the architecture revision. For architectures defined by Arm this is the minor revision.

For debug, the revision defined by Armv8 is 0x0.

All other values are reserved.

Reads as 0b0000.

Access to this field is **RO**.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component. Defined values are:

ARCHVER	Meaning
0b0110	Armv8 debug architecture.
0b0111	Armv8 debug architecture with Virtualization Host Extensions.
0b1000	Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.
0b1011	Armv8.9 debug architecture, FEAT_Debugv8p9.

EDDEVARCH.ARCHVER and EDDEVARCH.ARCHPART are also defined as a single field, EDDEVARCH.ARCHID, so that EDDEVARCH.ARCHVER is EDDEVARCH.ARCHID[15:12].

FEAT_VHE adds the functionality identified by the value 0b0111.

FEAT_Debugv8p2 adds the functionality identified by the value 0b1000.

FEAT_Debugv8p4 adds the functionality identified by the value 0b1001.

FEAT_Debugv8p8 adds the functionality identified by the value 0b1010.

FEAT_Debugv8p9 adds the functionality identified by the value 0b1011.

From Armv8.1, when FEAT_VHE is implemented the value 0b0110 is not permitted.

From Armv8.2, the values 0b0110 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 is not permitted.

From Armv8.9, the value 0b1010 is not permitted.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

ARCHPART	Meaning
0xA15	Armv8-A debug architecture.

EDDEVARCH.ARCHVER and EDDEVARCH.ARCHPART are also defined as a single field, EDDEVARCH.ARCHID, so that EDDEVARCH.ARCHPART is EDDEVARCH.ARCHID[11:0].

Armv8-A debug architecture.

Access to this field is **RO**.

Accessing EDDEVARCH

EDDEVARCH can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0xFBC	EDDEVARCH

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDDEVID, External Debug Device ID register 0

The EDDEVID characteristics are:

Purpose

Provides extra information for external debuggers about features of the debug implementation.

Configuration

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

EDDEVID is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				AuxRegs				RES0												DebugPower				PCSample							

Bits [31:28]

Reserved, RES0.

AuxRegs, bits [27:24]

Indicates support for Auxiliary registers. Defined values are:

AuxRegs	Meaning
0b0000	None supported.
0b0001	Support for External Debug Auxiliary Control Register, EDACR .

All other values are reserved.

Bits [23:8]

Reserved, RES0.

DebugPower, bits [7:4]

Indicates support for the FEAT_DoPD feature. Defined values are:

DebugPower	Meaning
0b0000	FEAT_DoPD not implemented. Registers in the external debug interface register map are implemented in a mix of the Debug and Core power domains.
0b0001	FEAT_DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.

FEAT_DoPD implements the functionality added by the value 0b0001.

All other values are reserved.

PCSample, bits [3:0]

Indicates the level of PC Sample-based Profiling support using external debug registers. Defined values are:

PCSample	Meaning
0b0000	PC Sample-based Profiling Extension is not implemented in the external debug registers space.
0b0010	Only EDPCSR and EDCIDSR are implemented. This option is only permitted if EL3 and EL2 are not implemented.
0b0011	EDPCSR , EDCIDSR , and EDVIDSR are implemented.

All other values are reserved.

When FEAT_PCSRv8p2 is implemented, the only permitted value is 0b0000.

Note

FEAT_PCSRv8p2 implements the PC Sample-based Profiling Extension in the Performance Monitors register space, as indicated by the value of [PMU.PMDEVID.PCSample](#). [PMDEVID.PCSample](#).

Accessing EDDEVID

EDDEVID can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0xFC8	EDDEVID

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDDEVID1, External Debug Device ID register 1

The EDDEVID1 characteristics are:

Purpose

Provides extra information for external debuggers about features of the debug implementation.

Configuration

- If FEAT_DoPD is implemented, this register is in the Core power domain.
- If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

EDDEVID1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								HSR			PCSROffset				

Bits [31:8]

Reserved, RES0.

HSR, bits [7:4]

Indicates support for the External Debug Halt Status Register, (EDHSR). Defined values are:

HSR	Meaning
0b0000	EDHSR is not implemented, and the PE follows behaviors consistent with all of the EDHSR fields having a zero value.
0b0001	EDHSR is implemented.
0b0010	As 0b0001, but extends EDHSR to include the VNCR, CM, and WnR fields.

All other values are reserved.

- When FEAT_SME and FEAT_Debugv8p9 are not implemented, the only permitted value is 0b0000.
- If FEAT_SME is implemented, the permitted values are 0b0000 and 0b0001.
- When If FEAT_Debugv8p9FEAT_SME is not implemented, the values only permitted value is 0b0000 and 0b0001 are not permitted.

PCSROffset, bits [3:0]

Indicates the offset applied to PC samples returned by reads of EDPCSR. Permitted values of this field in Armv8 are:

PCSRoffset	Meaning
0b0000	EDPCSR not implemented.
0b0010	EDPCSR implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state.

When FEAT_PCSRv8p2 is implemented, the only permitted value is 0b0000.

Note

FEAT_PCSRv8p2 implements the PC Sample-based Profiling Extension in the Performance Monitors register space, as indicated by the value of `PMU.PMDEVID.PCSample.PMDEVID.PCSample`.

Accessing EDDEVID1

EDDEVID1 can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0xFC4	EDDEVID1

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDDFR, External Debug Feature Register

The EDDFR characteristics are:

Purpose

Provides top level information about the debug system.

Note

Debuggers must use [EDDEVARCH](#) to determine the Debug architecture version.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

It is IMPLEMENTATION DEFINED whether EDDFR is implemented in the Core power domain or in the Debug power domain.

Attributes

EDDFR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
UNKNOWN				ExtTrcBuffRES0				UNKNOWN												TraceFilt				UNKNOWN							
CTX_CMPs				SEBEPRES0				WRPs				PMSSRES0				BRPs				PMUVer				TraceVer				UNKNOWN			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:60]

Reserved, UNKNOWN.

ExtTrcBuff, bits [59:56]

Trace Buffer External Mode Extension.

Reserved, RES0.

ExtTrcBuff	Meaning
0b0000	Trace Buffer Extension not implemented or Trace Buffer External Mode not implemented.
0b0001	Trace Buffer Extension implemented and Trace Buffer External Mode implemented.

All other values are reserved.

FEAT_TRBE_EXT implements the functionality identified by the value 0b0001.

Bits [55:44]

Reserved, UNKNOWN.

TraceFilt, bits [43:40]

Armv8.4 Self-hosted Trace Extension version. Defined values are:

TraceFilt	Meaning
0b0000	Armv8.4 Self-hosted Trace Extension is not implemented.
0b0001	Armv8.4 Self-hosted Trace Extension is implemented.

All other values are reserved.

FEAT_TRF implements the functionality added by 0b0001.

From Armv8.4, the permitted values are 0b0000 and 0b0001.

Bits [39:32]

Reserved, UNKNOWN.

CTX_CMPs, bits [31:28]

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

If FEAT_Debugv8p9 is implemented and 16 or more breakpoints that are context-aware are implemented, this field reads as 0b1111.

In an Armv8-A implementation that supports AArch64, this field returns the value of ID_AA64DFR0_EL1.CTX_CMPs.

SEBEP, bits [27:24]

This field either has the same value as ID_AA64DFR0_EL1.SESEBP or reads as zero.

Reserved, RES0.

WRPs, bits [23:20]

Access to this field is **RO**.

Number of watchpoints, minus 1.

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

If FEAT_Debugv8p9 is implemented and 16 or more watchpoints are implemented, this field reads as 0b1111.

In an Armv8-A implementation that supports AArch64, this field returns the value of ID_AA64DFR0_EL1.WRPs.

The value of 0b0000 is reserved.

PMSS, bits [19:16]

This field either has the same value as ID_AA64DFR0_EL1.PMSS or reads as zero.

Reserved, RES0.

BRPs, bits [15:12]

Access to this field is **RO**.

Number of breakpoints, minus 1.

Number of breakpoints, minus 1. The value of 0b0000 is reserved.

If FEAT_Debugv8p9 is implemented and 16 or more breakpoints are implemented, this field reads as 0b1111.

In an Armv8-A implementation that supports AArch64, this field returns the value of [ID_AA64DFR0_EL1](#).BRPs.

The This value field of has an 0b0000 IMPLEMENTATION DEFINED is reserved value.

PMUVer, bits [11:8]

Performance Monitors Extension version.

This field does not follow the standard ID scheme, but uses the alternative ID scheme described in 'Alternative ID scheme used for the Performance Monitors Extension version'

Defined values are:

PMUVer	Meaning
0b0000	Performance Monitors Extension not implemented.
0b0001	Performance Monitors Extension, PMUv3 implemented.
0b0100	PMUv3 for Armv8.1. As 0b0001, and adds support for: <ul style="list-style-type: none"> Extended 16-bit PMU.PMEVTYPER<n>_EL0.evtCount field. PMEVTYPER<n>_EL0.evtCount field. If EL2 is implemented, the MDCR_EL2.HPMD control.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds support for the PMMIR_EL1 register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the MDCR_EL2.HCCD control. If EL3 is implemented, the MDCR_EL3.SCCD control.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds support for: <ul style="list-style-type: none"> The PMU.PMCR_EL0.FZO and, if EL2 is implemented, PMCR_EL0.FZO and, if EL2 is implemented, MDCR_EL2.HPMFZO controls. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls.
0b1000	PMUv3 for Armv8.8. As 0b0111, and: <ul style="list-style-type: none"> Extends the Common event number space to include 0x0040 to 0x00BF and 0x4040 to 0x40BF. Removes the CONSTRAINED UNPREDICTABLE behaviors if a reserved or unimplemented PMU event number is selected.
0b1001	PMUv3 for Armv8.9. As 0b1000, and: <ul style="list-style-type: none"> Updates the definitions of existing PMU events. Adds support for the PMUSERENR_EL0.UEN control and the PMUACR_EL1 register. Adds support for the EDEC.R.PME control.
0b1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported. Arm does not recommend this value for new implementations.

All other values are reserved.

FEAT_PMUv3 implements the functionality identified by the value 0b0001.

FEAT_PMUv3p1 implements the functionality identified by the value 0b0100.

FEAT_PMUv3p4 implements the functionality identified by the value 0b0101.

FEAT_PMUv3p5 implements the functionality identified by the value 0b0110.

FEAT_PMUv3p7 implements the functionality identified by the value 0b0111.

FEAT_PMUv3p8 implements the functionality identified by the value 0b1000.

FEAT PMUv3p9 implements the functionality identified by the value 0b1001.

From Armv8.1, if FEAT_PMUv3 is implemented, the value 0b0001 is not permitted.

From Armv8.4, if FEAT_PMUv3 is implemented, the value 0b0100 is not permitted.

From Armv8.5, if FEAT_PMUv3 is implemented, the value 0b0101 is not permitted.

From Armv8.7, if FEAT_PMuV3 is implemented, the value 0b0110 is not permitted.

From Armv8.8, if FEAT_PMuV3 is implemented, the value 0b0111 is not permitted.

From Armv8.9, if FEAT_PMUv3 is implemented, the value 0b1000 is not permitted.

TraceVer, bits [7:4]

Trace support. Indicates whether System register interface to a trace unit is implemented. Defined values are:

TraceVer	Meaning
0b0000	Trace unit System registers not implemented.
0b0001	Trace unit System registers implemented.

All other values are reserved.

A value of 0b0000 only indicates that no System register interface to a trace unit is implemented. A trace unit might nevertheless be implemented without a System register interface.

In an Armv8-A implementation that supports AArch64, this field returns the value of [ID_AA64DFR0_EL1.TraceVer](#).

Bits [3:0]

Reserved, UNKNOWN.

Accessing EDDFR

EDDFR can be accessed through the external debug interface:

Component	Offset	Instance	Range
Debug	0xD28	EDDFR	31:0

This interface is accessible as follows:

- When `IsCorePowered()` and `!DoubleLockStatus()`, accesses to this register are **RO**.
- Otherwise, accesses to this register are **IMPDEF**.

Component	Offset	Instance	Range
Debug	0xD2C	EDDFR	63:32

This interface is accessible as follows:

- When `IsCorePowered()` and `!DoubleLockStatus()`, accesses to this register are **RO**.
- Otherwise, accesses to this register are **IMPDEF**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

EDDFR1, External Debug Feature Register 1

The EDDFR1 characteristics are:

Purpose

Provides top level information about the debug system in AArch64.

Configuration

If FEAT_DoPD is not implemented then it is IMPLEMENTATION DEFINED whether EDDFR1 is in the Core power domain or Debug power domain. If FEAT_DoPD is implemented then EDDFR1 is in the Core power domain.

Note

If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented.

Attributes

EDDFR1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0												EBEP				ITE				ABLE				PMICNTR				SPMU			
CTX_CMPS								WRPs								BRPs								SYSPMUID							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:52]

Reserved, RES0.

EBEP, bits [51:48]

This field either has the same value as [ID_AA64DFR1_EL1.EBEP](#) or reads as zero.

ITE, bits [47:44]

This field either has the same value as [ID_AA64DFR1_EL1.ITE](#) or reads as zero.

ABLE, bits [43:40]

Address Breakpoint Linking Extension. Defined values are:

ABLE	Meaning
0b0000	Address Breakpoint Linking Extension not implemented.
0b0001	Address Breakpoint Linking Extension implemented.

All other values are reserved.

FEAT_ABLE implements the functionality identified by the value 0b0001.

If FEAT_ABLE is not implemented, then the only permitted value is 0b0000.

From Armv9.4, when FEAT_ABLE is implemented, the value 0b0000 is not permitted.

PMICNTR, bits [39:36]

This field either has the same value as [ID_AA64DFR1_EL1.PMICNTR](#) or reads as zero.

SPMU, bits [35:32]

This field either has the same value as [ID_AA64DFR1_EL1.SPMU](#) or reads as zero.

CTX_CMPs, bits [31:24]

When FEAT_Debugv8p9 is implemented and EDDFR.CTX_CMPs == 0b1111:

Number of breakpoints that are context-aware, minus 1.

The value 0x00 means 16 breakpoints that are context-aware are implemented. Otherwise, this field is the number of breakpoints that are context-aware, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

The value of this field is never greater than [ID_AA64DFR1_EL1.BRPs](#).

Otherwise:

Reserved, RES0.

WRPs, bits [23:16]

When FEAT_Debugv8p9 is implemented and EDDFR.WRPs == 0b1111:

Number of watchpoints, minus 1.

The value 0x00 means 16 watchpoints are implemented. Otherwise, this field is the number of watchpoints, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

Otherwise:

Reserved, RES0.

BRPs, bits [15:8]

When FEAT_Debugv8p9 is implemented and EDDFR.BRPs == 0b1111:

Number of breakpoints, minus 1.

The value 0x00 means 16 breakpoints are implemented. Otherwise, this field is the number of breakpoints, minus 1.

The values 0x01 to 0x0F and 0x40 to 0xFF are reserved.

Otherwise:

Reserved, RES0.

SYSPMUID, bits [7:0]

This field either has the same value as [ID_AA64DFR1_EL1.SYSPMUID](#) or reads as zero.

Accessing EDDFR1

EDDFR1 can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0xD48	EDDFR1

Accesses to this interface are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDECR, External Debug Execution Control Register

The EDECR characteristics are:

Purpose

Controls Halting debug events.

Configuration

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

EDECR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																TRBE	SS	TRCERCE	PMEOSUCE	RES0	SS	RCE	OSUCE	RES0	SS	RCE	OSUCE	RES0	SS	RCE	OSUCE

Bits [31:7]3

Reserved, RES0.

TRBE, bit [6]

When FEAT_Debugv8p9 is implemented and FEAT_TRBE_EXT is implemented:

Trace Buffer External Debug Request Enable.

TRBE	Meaning
0b0	Trace Buffer External Debug Request disabled.
0b1	Trace Buffer External Debug Request enabled.

This bit is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

TRCE, bit [5]

When FEAT_ETEv1p3 is implemented and FEAT_Debugv8p9 is implemented:

ETE External Debug Request Enable.

TRCE	Meaning
0b0	ETE External Debug Request disabled.
0b1	ETE External Debug Request enabled.

This bit is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

PME, bit [4]

When FEAT_Debugv8p9 is implemented and FEAT_PMUv3p9 is implemented:

PMU Overflow External Debug Request Enable.

PME	Meaning
0b0	PMU Overflow External Debug Request disabled.
0b1	PMU Overflow External Debug Request enabled.

This bit is in the Core power domain.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [3]

Reserved, RES0.

SS, bit [2]

Halting step enable. Possible values of this field are:

SS	Meaning
0b0	Halting step debug event disabled.
0b1	Halting step debug event enabled.

If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is CONSTRAINED UNPREDICTABLE as described in 'Changing the value of EDECR.SS when not in Debug state'.

The reset behavior of this field is:

- On a Cold reset, when FEAT_DoPD is implemented, this field resets to 0.
- On an External debug reset, when FEAT_DoPD is not implemented, this field resets to 0.

RCE, bit [1]

When FEAT_DoPD is not implemented:

Reset Catch Enable.

RCE	Meaning
0b0	Reset Catch debug event disabled.
0b1	Reset Catch debug event enabled.

The reset behavior of this field is:

- On an External debug reset, this field resets to 0.

Otherwise:

Reserved, RES0.

OSUCE, bit [0]

When FEAT_DoPD is not implemented:

OS Unlock Catch Enable.

OSUCE	Meaning
0b0	OS Unlock Catch debug event disabled.
0b1	OS Unlock Catch debug event enabled.

The reset behavior of this field is:

- On an External debug reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Accessing EDECR

EDECRC can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x024	EDECR

This interface is accessible as follows:

- When (FEAT_DoPD is not implemented or IsCorePowered()) and SoftwareLockStatus(), accesses to this register are **RO**.
- When (FEAT_DoPD is not implemented or IsCorePowered()) and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDHSR, External Debug HaltingHalt SyndromeStatus Register

The EDHSR characteristics are:

Purpose

HoldsProvides syndromeDebug informationHalt forStatus a debug event.information.

Configuration

This register is present only when (FEAT_SME is implemented and an implementation implements EDHSR) or FEAT_Debugv8p9 is implemented. Otherwise, direct accesses to EDHSR are RES0.

EDHSR is in the Core power domain.

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to EDHSR are RES0.

This register is only valid when the PE is in Debug state and [EDSCR](#).STATUS is 0b101011, indicating a Watchpoint debug event. Otherwise, it has an UNKNOWN value.

The field [EDDEVID1](#).HSR indicates support for this register.

Attributes

EDHSR is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																
RES0								WPT				WPTV	WPF	FnP	RES0	VNCR	FnV	RES0	FnV	RES0	CM	RES0	WnR					RES0				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:24]

Reserved, RES0.

WPT, bits [23:18]

All other values are reserved.

Watchpoint number. When EDHSR.WPTV is 1number, holds0 theto index15 of a watchpoint that triggered the Watchpoint debug event.inclusive.

The reset behavior of this field is:

- On a WarmCold reset, this field resets to an architecturally UNKNOWN value.

WPTV, bit [17]

Watchpoint number valid.Valid.

WPTV	Meaning	Applies when
0b0	EDHSR.WPTV is not valid, and holds an UNKNOWN value.	When FEAT_SME is implemented and FEAT_Debugv8p9 is not implemented
0b1	EDHSR.WPTV is valid, and holds the number of a watchpoint that triggered the Watchpoint debug event. Debug state.	

When an entry to Debug state is triggered by a watchpoint match:

- If the PE sets any of FnV, FnP, or WPF to 1, then the PE sets WPTV to 1.
- If the PE sets all of FnV, FnP, and WPF to 0, then the PE sets WPTV to an IMPLEMENTATION DEFINED value, 0 or 1.

The reset behavior of this field is:

- On a WarmCold reset, this field resets to an architecturally UNKNOWN value.

WPF, bit [16]

Watchpoint match might be False.false-positive.

WPF	Meaning	Applies when
0b0	The watchpoint matched the original access or set of contiguous accesses.	
0b1	The watchpoint matched an access or set of contiguous accesses where the lowest accessed address was rounded down to the nearest multiple of 16 bytes and the highest accessed address was rounded up to the nearest multiple of 16 bytes minus 1, but the watchpoint might not have matched the original address of the access or set of contiguous accesses.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

The reset behavior of this field is:

- On a WarmCold reset, this field resets to an architecturally UNKNOWN value.

FnP, bit [15]

This field only has meaning if the EDWAR is valid; that is, when the FnV field is 0. If the FnV field is 1, the FnP field is 0.

EDWAR FAR not Precise.

FnP	Meaning	Applies when
0b0	If the FnV field is 0, the EDWAR holds the virtual address of an access or set of contiguous accesses that triggered an entry to Debug state. EDWAR is valid, it holds the virtual address of an access or sequence of contiguous accesses that triggered the Watchpoint debug event.	
0b1	If the EDWAR holds any address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered an entry to Debug state. EDWAR is valid, it holds any virtual address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered the Watchpoint debug event.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

The reset behavior of this field is:

- On a WarmCold reset, this field resets to an architecturally UNKNOWN value.

BitBits [14:11]

Reserved, RES0.

VNCR, bit [13]

When FEAT_NV2 is implemented and FEAT_Debugv8p9 is implemented:

VNCR_EL2 access. Indicates that the Watchpoint debug event came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning
0b0	The Watchpoint debug event was not generated by the use of VNCR_EL2 by EL1 code.
0b1	The Watchpoint debug event was generated by the use of VNCR_EL2 by EL1 code.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

FnV, bit [10]

EDWAR FAR not Valid.

FnV	Meaning	Applies when
0b0	EDWAR The EDWAR is valid, and its value is as described by the FnP field.	
0b1	EDWAR The EDWAR is not valid, and holds an UNKNOWN value.	When (FEAT_SVE is implemented and FEAT_Debugv8p9 is implemented) or FEAT_SME is implemented

The reset behavior of this field is:

- On a WarmCold reset, this field resets to an architecturally UNKNOWN value.

BitBits [9:0]

Reserved, RES0.

CM, bit [8]

When FEAT_Debugv8p9 is implemented:

Cache maintenance. Indicates whether the Watchpoint debug event came from a cache maintenance instruction.

CM	Meaning
0b0	The Watchpoint debug event was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Watchpoint debug event was generated by the execution of a cache maintenance instruction. The DC ZVA , DC GVA , and DC GZVA instructions are not cache maintenance instructions, and therefore do not cause this bit to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [7]

Reserved, RES0.

WnR, bit [6]

When FEAT_Debugv8p9 is implemented:

Write not Read. Indicates whether the Watchpoint debug event was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Watchpoint debug event caused by an instruction reading from a memory location.
0b1	Watchpoint debug event caused by an instruction writing to a memory location.

For Watchpoint debug events on cache maintenance instructions, this bit is set to 1.

For Watchpoint debug events from an atomic instruction, this bit is set to 0 if a read of the location would have generated the Watchpoint debug event, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is UNPREDICTABLE which watchpoint generates the Watchpoint debug event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [5:0]

Reserved, RES0.

Accessing EDHSR

EDHSR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x038	EDHSR

Component	Offset	Instance	Range
Debug	0x038	EDHSR	31:0

This interface is accessible as follows:

- When DoubleLockStatus(), or !IsCorePowered() or OSLockStatus(), accesses to this register generate an error response.
- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register are generate an error response. **RO**.

Component	Offset	Instance	Range
Debug	0x03C	EDHSR	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDPCSR, External Debug Program Counter Sample Register

The EDPCSR characteristics are:

Purpose

Holds a sampled instruction address value.

Configuration

EDPCSR is in the Core power domain.

This register is present only when FEAT_PCSRv8 is implemented and FEAT_PCSRv8p2 is not implemented. Otherwise, direct accesses to EDPCSR are RES0.

EDPCSR[63:32] and EDPCSR[31:0] are accessed at 32-bit memory mapped addresses that are not contiguous.

If FEAT_VHE is implemented, the format of this register differs depending on the value of [EDSCR.SC2](#).

Implemented only if the OPTIONAL PC Sample-based Profiling Extension is implemented in the external debug registers space.

Note

FEAT_PCSRv8p2 implements the PC Sample-based Profiling Extension in the Performance Monitors registers space.

Attributes

EDPCSR is a 64-bit register.

Field descriptions

When FEAT_VHE is not implemented or EDSCR.SC2 == 0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
PC Sample high word, EDPCSRhi																															
PC Sample low word																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

PC Sample high word, EDPCSRhi. If [EDVIDSR.HV](#) == 0 then this field is RAZ, otherwise bits [63:32] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR.{NS,E2,E3}](#).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [31:0]

PC Sample low word. EDPCSRlo, bits[31:0] of the sampled instruction address value.

EDPCSRlo reads as 0xFFFFFFFF when any of the following are true:

- The PE is in Debug state.
- PC Sample-based profiling is prohibited.

If a branch instruction has retired since the PE left reset state, then the first read of EDPCSR[31:0] is permitted but not required to return 0xFFFFFFFF.

EDPCSRlo reads as an UNKNOWN value when any of the following are true:

- The PE is in reset state.
- No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited.
- No branch instruction has retired since the last read of EDPCSR[31:0].

For the cases where a read of EDPCSR[31:0] returns 0xFFFFFFFF or an UNKNOWN value, the read has the side-effect of setting EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) to UNKNOWN values.

Otherwise, a read of EDPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#). The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.

For a read of EDPCSR[31:0] from the memory-mapped interface, if EDLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the side-effect of the access does not occur and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) are unchanged.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When FEAT_VHE is implemented and EDSCR.SC2 == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	EL	RES0						PC Sample high word, EDPCSRhi																								
PC Sample low word																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63]

Non-secure state sample. Indicates the Security state that is associated with the most recent EDPCSR sample or, when it is read as a single atomic 64-bit read, the current EDPCSR sample. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

If EL3 is not implemented, this bit indicates the Effective value of SCR.NS.

NS	Meaning
0b0	Sample is from Secure state.
0b1	Sample is from Non-secure state.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

EL, bits [62:61]

Exception level status sample. Indicates the Exception level that is associated with the most recent EDPCSR sample or, when it is read as a single atomic 64-bit read, the current EDPCSR sample. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

EL	Meaning
0b00	Sample is from EL0.
0b01	Sample is from EL1.
0b10	Sample is from EL2.
0b11	Sample is from EL3.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [60:56]

Reserved, RES0.

Bits [55:32]

PC Sample high word, EDPCSRhi. Bits [55:32] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [31:0]

PC Sample low word. EDPCSRlo, bits[31:0] of the sampled instruction address value.

EDPCSRlo reads as 0xFFFFFFFF when any of the following are true:

- The PE is in Debug state.
- PC Sample-based profiling is prohibited.

If a branch instruction has retired since the PE left reset state, then the first read of EDPCSR[31:0] is permitted but not required to return 0xFFFFFFFF.

EDPCSRlo reads as an UNKNOWN value when any of the following are true:

- The PE is in reset state.
- No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited.
- No branch instruction has retired since the last read of EDPCSR[31:0].

For the cases where a read of EDPCSR[31:0] returns 0xFFFFFFFF or an UNKNOWN value, the read has the side-effect of setting EDPCSRhi, [EDCIDS](#)R, and [EDVIDS](#)R to UNKNOWN values.

Otherwise, a read of EDPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to EDPCSRhi, [EDCIDS](#)R, and [EDVIDS](#)R. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

For a read of EDPCSR[31:0] from the memory-mapped interface, if EDLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the side-effect of the access does not occur and EDPCSRhi, [EDCIDS](#)R, and [EDVIDS](#)R are unchanged.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing EDPCSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'

EDPCSR can be accessed through the external memory-mapped debug interface interfaces:

Component	Offset	Instance	Range
Debug	0x0A0	EDPCSR	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
-----------	--------	----------	-------

Debug	0x0AC	EDPCSR	63:32
-------	-------	--------	-------

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

EDSCR, External Debug Status and Control Register

The EDSCR characteristics are:

Purpose

Main control register for the debug implementation.

Configuration

External register EDSCR bits [30:29] are architecturally mapped to AArch64 System register [MDCCSR_EL0\[30:29\]](#).

EDSCR is in the Core power domain.

Attributes

EDSCR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TF	ORXfull	TXfull	IT	ORXO	TXU	PipeAdv	ITE	INTdis	TDAMA	SC2NS	RES0	SDD	NSE	HDE		RW		EL	AERR	STATUS											

TFO, bit [31]

When FEAT_TRF is implemented:

Trace Filter Override. Overrides the Trace Filter controls allowing the external debugger to trace any visible Exception level.

TFO	Meaning
0b0	Trace Filter controls are not affected.
0b1	Trace Filter controls in TRFCR_EL1 and TRFCR_EL2 are ignored.
	Trace Filter controls TRFCR and HTRFCR are ignored.

When [OSLSR_EL1](#).OSLK is 1, this bit can be indirectly read and written through the following System registers:

- [MDSCR_EL1](#).
- [DBGDSCRext](#).

This bit is ignored by the PE when any of the following is true:

- ExternalSecureNoninvasiveDebugEnabled() is FALSE and the Effective value of [MDCR_EL3](#).STE is 1.
- FEAT_RME is implemented, ExternalRealmNoninvasiveDebugEnabled() is FALSE, and the Effective value of [MDCR_EL3](#).RLTE is 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

RXfull, bit [30]

DTRRX full.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Access to this field is **RO**.

TXfull, bit [29]

DTRTX full.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Access to this field is **RO**.

ITO, bit [28]

ITR overrun. Set to 0 on entry to Debug state.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

RXO, bit [27]

DTRRX overrun.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Access to this field is **RO**.

TXU, bit [26]

DTRTX underrun.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Access to this field is **RO**.

PipeAdv, bit [25]

Pipeline Advance. Indicates that software execution is progressing.

PipeAdv	Meaning
0b0	No progress has been made by the PE since the last time this field was cleared to zero by writing 1 to EDRCR.CSPA .
0b1	Progress has been made by the PE since the last time this field was cleared to zero by writing 1 to EDRCR.CSPA .

The architecture does not define precisely when this field is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing. For example, a PE might set this field to 1 each time the PE retires one or more instructions, or at periodic intervals during the progression of an instruction.

When FEAT_MOPS is implemented, CPY, CPYF, SET, and SETG are examples of instructions that periodically make forward progress.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Access to this field is **RO**.

ITE, bit [24]

ITR empty.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

INTdis, bits [23:22]

When FEAT_RME is implemented:

Interrupt disable. Disables taking interrupts in Non-debug state.

INTdis	Meaning
0b00	This bit has no effect on the masking of interrupts.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked. If ExternalRootInvasiveDebugEnabled() is TRUE, then all interrupts taken to Root state are masked. If ExternalRealmInvasiveDebugEnabled() is TRUE, then all interrupts taken to Realm state are masked.

Note

All interrupts includes virtual and SError interrupts.

When [OSLSR_EL1](#).OSLK is 1, this field can be indirectly read and written through the following System registers:

- [MDSCR_EL1](#).
- [DBGDSCRext](#).

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

When FEAT_RME is implemented, bit[23] of this register is RES0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

When FEAT_Debugv8p4 is implemented:

Interrupt disable. Disables taking interrupts in Non-debug state.

INTdis	Meaning
0b00	Masking of interrupts is controlled by PSTATE and interrupt routing controls.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked.

Note

All interrupts includes virtual and SError interrupts.

When [OSLSR_EL1](#).OSLK is 1, this field can be indirectly read and written through the following System registers:

- [MDSCR_EL1](#).
- [DBGDSCRext](#).

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

When FEAT_Debugv8p4 is implemented, bit[23] of this register is RES0.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Interrupt disable. Disables taking interrupts in Non-debug state.

INTdis	Meaning
0b00	Masking of interrupts is controlled by PSTATE and interrupt routing controls.
0b01	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure EL1 are masked.
0b10	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure EL1 are masked.
0b11	If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked.

Note

All interrupts includes virtual and SError interrupts.

When [OSLSR_EL1](#).OSLK is 1, this field can be indirectly read and written through the following System registers:

- [MDSCR_EL1](#).
- [DBGDSCRext](#).

The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.

Support for the values 0b01 and 0b10 is IMPLEMENTATION DEFINED. If these values are not supported, they are reserved. If programmed with a reserved value, the PE behaves as if INTdis has been programmed with a defined value, other than for a direct read of EDSCR, and the value returned by a read of EDSCR.INTdis is UNKNOWN.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

TDA, bit [21]

Traps accesses to the following debug System registers:

- AArch64: [DBGBCR<n>_EL1](#), [DBGBVR<n>_EL1](#), [DBGWCR<n>_EL1](#), [DBGWVR<n>_EL1](#).
- AArch32: [DBGBCR<n>](#), [DBGBVR<n>](#), [DBGBXVR<n>](#), [DBGWCR<n>](#), [DBGWVR<n>](#).

TDA	Meaning
0b0	Accesses to debug System registers do not generate a Software Access Debug event.
0b1	Accesses to debug System registers generate a Software Access Debug event, if OSLSR_EL1 .OSLK is 0 and if halting is allowed.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

MA, bit [20]

Memory access mode. Controls the use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state.

Possible values of this field are:

MA	Meaning
0b0	Normal access mode.
0b1	Memory access mode.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

SC2, bit [19]

When FEAT_PCSRv8 is implemented, (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented) and FEAT_PCSRv8p2 is not implemented:

Sample [CONTEXTIDR_EL2](#). Controls whether the PC Sample-based Profiling Extension samples [CONTEXTIDR_EL2](#) or [VTTBR_EL2](#).VMID.

SC2	Meaning
0b0	Sample VTTBR_EL2 .VMID.
0b1	Sample CONTEXTIDR_EL2 .

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

NS, bit [18]

When FEAT_RME is implemented:

Non-secure status. Together with the NSE field, gives the current Security state:

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

Otherwise:

Non-secure status. In Debug state, gives the current Security state:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **UNKNOWN/WI**.

- Otherwise, access to this field is **RO**.

Bit [17]

Reserved, RES0.

SDD, bit [16]**When FEAT_RME is implemented:**

EL3 debug disabled.

On entry to Debug state:

- If entering from EL3, SDD is set to 0.
- Otherwise, SDD is set to the inverse of `ExternalRootInvasiveDebugEnabled()`.

In Debug state, the value of SDD does not change, even if `ExternalRootInvasiveDebugEnabled()` changes.

In Non-debug state, SDD returns the inverse of `ExternalRootInvasiveDebugEnabled()`.

Access to this field is **RO**.

Otherwise:

Secure debug disabled.

On entry to Debug state:

- If entering in Secure state, SDD is set to 0.
- If entering in Non-secure state, SDD is set to the inverse of `ExternalSecureInvasiveDebugEnabled()`.

In Debug state, the value of the SDD bit does not change, even if `ExternalSecureInvasiveDebugEnabled()` changes.

In Non-debug state:

- SDD returns the inverse of `ExternalSecureInvasiveDebugEnabled()`. If the authentication signals that control `ExternalSecureInvasiveDebugEnabled()` change, a context synchronization event is required to guarantee their effect.
- This bit is unaffected by the Security state of the PE.

If EL3 is not implemented and the implementation is Non-secure, this bit is RES1.

Access to this field is **RO**.

NSE, bit [15]**When FEAT_RME is implemented:**

Together with the NS field, this field gives the current Security state.

For a description of the values derived by evaluating NS and NSE together, see `EDSCR.NS`.

In Non-debug state, this bit is UNKNOWN.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

HDE, bit [14]

Halting debug enable.

HDE	Meaning
0b0	Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.
0b1	Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

RW, bits [13:10]

Exception level Execution state status. In Debug state, each bit gives the current Execution state of each Exception level.

RW	Meaning	Applies when
0b1111	Any of the following: <ul style="list-style-type: none"> The PE is in Non-debug state. The PE is at EL0 using AArch64. The PE is not at EL0, and EL1, EL2, and EL3 are using AArch64. 	
0b1110	The PE is in Debug state at EL0. EL0 is using AArch32. EL1, EL2, and EL3 are using AArch64.	When AArch32 is supported
0b110x	The PE is in Debug state. EL0 and EL1 are using AArch32. EL2 is enabled in the current Security state and is using AArch64. If implemented, EL3 is using AArch64.	When AArch32 is supported and EL2 is implemented
0b10xx	The PE is in Debug state. EL0 and EL1 are using AArch32. EL2 is not implemented, disabled in the current Security state, or using AArch32. EL3 is using AArch64.	When AArch32 is supported and EL3 is implemented
0b0xxx	The PE is in Debug state. All Exception levels are using AArch32.	When AArch32 is supported

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **RAO/WI**.
- Otherwise, access to this field is **RO**.

EL, bits [9:8]

Exception level. In Debug state, gives the current Exception level of the PE.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

A, bit [7]

SError interrupt pending. In Debug state, indicates whether an SError interrupt is pending:

- If [HCR_EL2](#).{AMO, TGE} = {1, 0}, EL2 is enabled in the current Security state, and the PE is executing at EL0 or EL1, a virtual SError interrupt.
- Otherwise, a physical SError interrupt.

A	Meaning
0b0	No SError interrupt pending.
0b1	SErrror interrupt pending.

A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted.

Accessing this field has the following behavior:

- When the PE is in Non-debug state, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

ERR, bit [6]

Cumulative error flag. This bit is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Access to this field is **RO**.

STATUS, bits [5:0]

Debug status flags.

STATUS	Meaning
0b000001	PE is restarting, exiting Debug state.
0b000010	PE is in Non-debug state.
0b000111	Breakpoint.
0b010011	External debug request.
0b011011	Halting step, normal.
0b011111	Halting step, exclusive.
0b100011	OS Unlock Catch.
0b100111	Reset Catch.
0b101011	Watchpoint.
0b101111	HLT instruction.
0b110011	Software access to debug register.
0b110111	Exception Catch.
0b111011	Halting step, no syndrome.

All other values of STATUS are reserved.

Access to this field is **RO**.

Accessing EDSCR

EDSCR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x088	EDSCR

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

(old)

htmldiff from-

(new)

EDSCR2, External Debug Status and Control Register 2

The EDSCR2 characteristics are:

Purpose

Main control register 2 for the debug implementation.

Configuration

External register EDSCR2 bits [31:0] are architecturally mapped to AArch64 System register [MDSCR_EL1\[63:32\]](#).

This register is present only when FEAT_Debugv8p9 is implemented or FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to EDSCR2 are RES0.

EDSCR2 is in the Core power domain.

Attributes

EDSCR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																TTA		EBWE													

Bits [31:2]

Reserved, RES0.

TTA, bit [1]

When FEAT_TRBE_EXT is implemented or FEAT_ETEv1p3 is implemented:

Trap Trace Accesses.

Traps access to the following System registers:

AArch64: [TRBBASER_EL1](#), [TRBLIMITR_EL1](#), [TRBMAR_EL1](#), [TRBPTR_EL1](#), [TRBSR_EL1](#), [TRBTRG_EL1](#), [TRCACATR<n>](#), [TRCACVR<n>](#), [TRCAUTHSTATUS](#), [TRCAUXCTLR](#), [TRCBBCTLR](#), [TRCCCCTLR](#), [TRCCIDCCTLR0](#), [TRCCIDCCTLR1](#), [TRCCIDCVR<n>](#), [TRCCCLAIMCLR](#), [TRCCCLAIMSET](#), [TRCCNTCTLR<n>](#), [TRCCNTRLDVR<n>](#), [TRCCNTVR<n>](#), [TRCCONFIGR](#), [TRCDEVARCH](#), [TRCDEVID](#), [TRCEVENTCTL0R](#), [TRCEVENTCTL1R](#), [TRCEXTINSELR<n>](#), [TRCIDR0](#), [TRCIDR1](#), [TRCIDR2](#), [TRCIDR3](#), [TRCIDR4](#), [TRCIDR5](#), [TRCIDR6](#), [TRCIDR7](#), [TRCIDR8](#), [TRCIDR9](#), [TRCIDR10](#), [TRCIDR11](#), [TRCIDR12](#), [TRCIDR13](#), [TRCIMSPEC0](#), [TRCIMSPEC<n>](#), [TRCITEEDCR](#), [TRCOSLSR](#), [TRCPRGCTLR](#), [TRCQCTLR](#), [TRCRSCTLR<n>](#), [TRCRSR](#), [TRCSEQEVR<n>](#), [TRCSEQORSTEV](#), [TRCSEQSTR](#), [TRCSSCCR<n>](#), [TRCSSCSR<n>](#), [TRCSSPCICR<n>](#), [TRCSTALLCTLR](#), [TRCSTATR](#), [TRCSYNCPR](#), [TRCTRACEIDR](#), [TRCTSCTLR](#), [TRCVICTLR](#), [TRCVIIECTLR](#), [TRCVIPCSSCTLR](#), [TRCVISSCTLR](#), [TRCVIMDCCTLR0](#), [TRCVIMDCCTLR1](#), and [TRCVIMDCVR<n>](#).

TTA	Meaning
0b0	Accesses to trace System registers do not generate a Software Access debug event.
0b1	Accesses to trace System registers generate a Software Access debug event, if OSLSR_EL1 .OSLK is 0 and if halting is allowed.

When [OSLSR_EL1](#).OSLK is 1, this bit can be read and written through the [MDSCR_EL1](#) System register.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

EBWE, bit [0]

When FEAT_Debugv8p9 is implemented:

Extended Breakpoint and Watchpoint Enable. Enables use of additional breakpoints or watchpoints.

EBWE	Meaning
0b0	Breakpoints and watchpoints above 15 are disabled.
0b1	Breakpoints and watchpoints above 15 are not affected by this field.

It is IMPLEMENTATION DEFINED whether this field is implemented or is RES0 when 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and MDSELR_EL1 is implemented as RAZ/WI.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Accessing EDSCR2

EDSCR2 can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x028	EDSCR2

This interface is accessible as follows:

- When DoubleLockStatus(), or !IsCorePowered() or OSLockStatus(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

ERRDEVARCH, Device Architecture Register

The ERRDEVARCH characteristics are:

Purpose

Provides discovery information for the component.

Configuration

ERRDEVARCH is implemented only as part of a memory-mapped group of error records.

Attributes

ERRDEVARCH is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											

ARCHITECT, bits [31:21]

Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.

ARCHITECT	Meaning
0b01000111011	JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.

This field reads as 0x23B.

AccessOther to values thisare fielddefined isby the JEDEC JEP106 standard. RO.

PRESENT, bit [20]

DEVARCH present.Present. Defines that ERRDEVARCHthe DEVARCH register is present.

PRESENT	Meaning
0b0	Device Architecture information not present.
0b1	Device Architecture information present.

This field reads as 1.

REVISION, bits [19:16]

When UInt(ERRDEVARCH.ARCHPART) == 0xA00 and ERRDEVARCH.ARCHVER == 0b0000:

Revision. Defines the architecture revision of the component.

REVISION	Meaning
0b0000	RAS System Architecture, error record group v1.0.
0b0001	RAS System Architecture, error record group v1.1. As 0b0000 and also: <ul style="list-style-type: none"> • Simplifies ERR<n>STATUS. • Adds support for additional ERR<n>MISC<m> registers. • Adds support for the optional RAS Timestamp Extension. • Adds support for the optional Common Fault Injection Model Extension.

All other values are reserved.

When UInt(ERRDEVARCH.ARCHPART) == 0xA00 and ERRDEVARCH.ARCHVER == 0b0001:

ARCHVER, bits [15:12]

~~Revision. Architecture Version.~~ Defines the architecture ~~revision~~version of the component.

REVISIONARCHVER	Meaning
0b0000	RAS System Architecture, error record group v2.0.v1.

All other values are reserved.

~~This field reads as 0b0000.~~

~~AccessARCHVER to and thisARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. RO.~~

When UInt(ERRDEVARCH.ARCHPART) == 0xA08 and ERRDEVARCH.ARCHVER == 0b0000:

Revision. Defines the architecture revision of the component.

REVISION	Meaning
0b0000	RAS System Architecture, fault injection group v1.0.

All other values are reserved.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

ARCHVER, bits [15:12]

When UInt(ERRDEVARCH.ARCHPART) == 0xA00:

Architecture Version. Defines the architecture version of the component.

ARCHVER	Meaning
0b0000	RAS System Architecture, error record group v1.
0b0001	RAS System Architecture, error record group v2. As 0b0000 and also defines fields in ERRDEVID that describe additional properties of this error record group.

All other values are reserved.

ERRDEVARCH.ARCHVER and ERRDEVARCH.ARCHPART are also defined as a single field, ERRDEVARCH.ARCHID, so that ERRDEVARCH.ARCHVER is ERRDEVARCH.ARCHID[15:12].

(old)

htmldiff from-

(new)

ERRDEVID, Device Configuration Register

The ERRDEVID characteristics are:

Purpose

Provides discovery information for the component.

Configuration

ERRDEVID is implemented only as part of a memory-mapped group of error records.

Attributes

ERRDEVID is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										PFG		NUM		AGENT		MSI		NUM													

Bits [31:22:16]

Reserved, RES0.

PFG, bit [21]

When RAS System Architecture v2 is implemented:

Common Fault Injection Mechanism. Describes whether any Common Fault Injection Mechanism registers are implemented in the same page as this register.

PFG	Meaning
0b0	Any Common Fault Injection Mechanism registers are implemented in the same page as this register.
0b1	Any Common Fault Injection Mechanism registers are implemented in a separate fault injection group page.

Accessing this field has the following behavior:

- When ERRDEVID is part of a fault injection group, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

Otherwise:

Reserved, RAZ.

AGENT, bit [20]

When RAS System Architecture v2 is implemented:

RAS agent. Describes whether the error record group containing ERRDEVID is part of a RAS agent that is not a System RAS agent.

AGENT	Meaning
0b0	The error record group containing ERRDEVID is a System RAS agent. That is, it directly generates any supported RAS interrupts.
0b1	The error record group containing ERRDEVID is not a System RAS agent. That is, it generates any supported RAS interrupts by signaling them to another RAS agent.

Accessing this field has the following behavior:

- When ERRDEVID is part of a fault injection group, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

Otherwise:

Reserved, RAZ.

MSI, bits [19:16]

Message Signaled Interrupt registers. Describes whether the MSI registers are implemented.

MSI	Meaning
0b0000	It is IMPLEMENTATION DEFINED whether any MSI registers are implemented.
0b0001	An IMPLEMENTATION DEFINED form of MSI registers are implemented.
0b0010	The recommended layout form of MSI registers are implemented.
0b1111	MSI registers are not implemented.

All other values are reserved.

Accessing this field has the following behavior:

- When ERRDEVID is part of a RAS agent that is not a System RAS agent, access to this field is **RAO/WI**.
- When ERRDEVID is part of a fault injection group, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

NUM, bits [15:0]

Highest numbered index of the error records in this group, plus one. Each implemented record is owned by a node. A node might own multiple records.

This manual describes a group of error records accessed via a standard 4KB memory-mapped peripheral. For a 4KB peripheral, up to 24 error records can be accessed if the Common Fault Injection Model is implemented, and up to 56 otherwise.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing ERRDEVID

ERRDEVID can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
RAS	0xFC8	ERRDEVID

Accesses to this interface are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

The ERR<n>CTLR characteristics are:

Purpose

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for uncorrected errors.

For each bit, if the node does not support the feature, then the bit is RES0. The definition of each record is IMPLEMENTATION DEFINED.

Configuration

This register is present only when error record <n> is implemented and error record <n> is the first error record owned by a node. Otherwise, direct accesses to ERR<n>CTRL are RES0.

ERR<n>FR describes the features implemented by the node.

Attributes

ERR<n>CTLR is a 64-bit register.

Field descriptions

63626160595857565554535251504948															47	46	45	44	43	42	41	40								
															IMPLEMENTATION DEFINE															
RES0															WDFI	CI	Bit[14]	RES0	CI	WDUI	CE	Bit[10]	WDUI	WCFI	Bit[10]	Bit[8]	WCFI	WUE	Bit[8]	WFI
31302928272625242322212019181716															15	14	13	12	11	10	9	8								

IMPLEMENTATION DEFINED, bits [63:32]

Reserved for IMPLEMENTATION DEFINED controls. Must permit SBZP write policy for software.

Bits [31:16]

Reserved, RES0.

WDFI, bit [15]

When RAS System Architecture v2 is implemented and ERR<n>FR.DFI == 0b11:

Fault handling interrupt for Deferred errors on writes enable, with ERR<n>CTLR.WFI.

When enabled by ERR<n>CTRL.{WDFI, WFI}:

- The fault handling interrupt is generated for errors recorded as Deferred error on writes.
- If the corresponding fault handling interrupt control for corrected error events, ERR<n>CTLR.WCFI, is not implemented, then the fault handling interrupt is generated for corrected error events on writes.

WDFI	Meaning
0b0	When ERR<n>CTLR.WFI == 0: Fault handling interrupt not generated for Deferred errors on writes. When ERR<n>CTLR.WFI == 1: Fault handling interrupt generated for Deferred errors on writes.
0b1	When ERR<n>CTLR.WFI == 0: Fault handling interrupt generated for Deferred errors on writes. When ERR<n>CTLR.WFI == 1: Fault handling interrupt not generated for Deferred errors on writes.

See ERR<n>CTLR.CFI for more information on corrected error events.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[14]

When RAS System Architecture v2 is implemented and ERR<n>FR.DFI == 0b10:

DFI, bit [0] of bit [14]

Fault handling interrupt for Deferred errors enable, with ERR<n>CTLR.FI.

When ERR<n>FR.DFI == 0b10, this control applies to errors on both reads and writes.

When enabled by ERR<n>CTLR.{DFI, FI}:

- The fault handling interrupt is generated for all errors recorded as Deferred error.
- If the fault handling interrupt control for corrected error events, ERR<n>CTLR.CFI, is not implemented, then the fault handling interrupt is generated for all corrected error events.

DFI	Meaning
0b0	When ERR<n>CTLR.FI == 0: Fault handling interrupt not generated for Deferred errors. When ERR<n>CTLR.FI == 1: Fault handling interrupt generated for Deferred errors.
0b1	When ERR<n>CTLR.FI == 0: Fault handling interrupt generated for Deferred errors. When ERR<n>CTLR.FI == 1: Fault handling interrupt not generated for Deferred errors.

See ERR<n>CTLR.CFI for more information on corrected error events.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When RAS System Architecture v2 is implemented and ERR<n>FR.DFI == 0b11:

RDFI, bit [0] of bit [14]

Fault handling interrupt for Deferred errors on reads enable, with ERR<n>CTLR.RFI.

When ERR<n>FR.DFI == 0b11, this field is named RDFI.

When enabled by ERR<n>CTLR.{RDFI, RFI}:

- The fault handling interrupt is generated for errors recorded as Deferred error on reads.
- If the corresponding fault handling interrupt control for corrected error events, ERR<n>CTLR.RCFI, is not implemented, then the fault handling interrupt is generated for corrected error events on reads.

RDFI	Meaning
0b0	When ERR<n>CTLR.RFI == 0: Fault handling interrupt not generated for Deferred errors on reads. When ERR<n>CTLR.RFI == 1: Fault handling interrupt generated for Deferred errors on reads.
0b1	When ERR<n>CTLR.RFI == 0: Fault handling interrupt generated for Deferred errors on reads. When ERR<n>CTLR.RFI == 1: Fault handling interrupt not generated for Deferred errors on reads.

See ERR<n>CTLR.CFI for more information on corrected error events.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CI, bit [13]

When ERR<n>FR.CI == 0b10:

Critical error interrupt enable. When enabled, the critical error interrupt is generated for a critical error condition.

CI	Meaning
0b0	Critical error interrupt not generated for critical errors. Critical errors are treated as Uncontained errors.
0b1	Critical error interrupt generated for critical errors.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CED, bit [12]

When RAS System Architecture v2 is implemented, ERR<n>FR.CEC != 0b000 and ERR<n>FR.CED == 1:

Disable error counting.

CED	Meaning
0b0	Error counter or counters enabled.
0b1	Error counter or counters disabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

WDUI, bit [11]**When ERR<n>FR.DUI == 0b11:**

Error recovery interrupt for Deferred errors on writes enable.

When enabled, the error recovery interrupt is generated for errors recorded as Deferred error on writes.

WDUI	Meaning
0b0	Error recovery interrupt not generated for Deferred errors on writes.
0b1	Error recovery interrupt generated for Deferred errors on writes.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[10]**When ERR<n>FR.DUI == 0b10:****DUI, bit [0] of bit [10]**

Error recovery interrupt for Deferred errors enable.

When [ERR<n>FR.DUI == 0b10](#), this control applies to errors arising from both reads and writes.

When enabled, the error recovery interrupt is generated for all errors recorded as Deferred error.

DUI	Meaning
0b0	Error recovery interrupt not generated for Deferred errors.
0b1	Error recovery interrupt generated for Deferred errors.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When ERR<n>FR.DUI == 0b11:**RDUI, bit [0] of bit [10]**

Error recovery interrupt for Deferred errors on reads enable.

When [ERR<n>FR.DUI == 0b11](#), this field is named RDUI.

When enabled, the error recovery interrupt is generated for errors recorded as Deferred error on reads.

RDUI	Meaning
0b0	Error recovery interrupt not generated for Deferred errors on reads.
0b1	Error recovery interrupt generated for Deferred errors on reads.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

WCFI, bit [9]

When ERR<n>FR.CFI == 0b11:

Fault handling interrupt for Corrected errors on writes enable.

When enabled:

- If the node implements Corrected error counters for writes, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see [ERR<n>MISC0](#).
- Otherwise, the fault handling interrupt is also generated for errors recorded as Corrected error on writes.

WCFI	Meaning
0b0	Fault handling interrupt not generated for Corrected errors on writes.
0b1	Fault handling interrupt generated for Corrected errors on writes.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[8]

When ERR<n>FR.CFI == 0b10:

CFI, bit [0] of bit [8]

- If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see [ERR<n>MISC0](#).
- Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error.

Fault handling interrupt for ~~corrected~~Corrected error eventserrors enable.

When [ERR<n>FR.CFI](#) == 0b10, this control applies to errors ~~on arising from~~ both reads and writes.

When enabled, the fault handling interrupt is generated for all corrected error events.:

CFI	Meaning
0b0	Fault handling interrupt not generated for corrected Corrected error events.errors.
0b1	Fault handling interrupt generated for corrected Corrected error events.errors.

If the node implements a corrected error counter or counters, and the counter or counters are enabled, then a corrected error event is defined as follows:

- A corrected error event occurs when a counter overflows and sets a counter overflow flag to 1.
- It is UNPREDICTABLE whether a corrected error event occurs when a software write sets a counter overflow flag to 1.
- It is UNPREDICTABLE whether a corrected error event occurs when a counter overflows and the overflow flag was previously set to 1.

Otherwise, a corrected error event occurs when the error record records an error as a Corrected error.

If the node implements a corrected error counter or counters and ERR<n>CTLR.CED is not implemented, then the counter or counters are always enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When ERR<n>FR.CFI == 0b11:

RCFI, bit [0] of bit [8]

Fault handling interrupt for ~~corrected~~Corrected error events.errors on reads enable.

When [ERR<n>FR.CFI](#) == 0b11, this field is named RCFI.

- ~~If the node implements Corrected error counters for reads, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see [ERR<n>MISC0](#).~~
- ~~Otherwise, the fault handling interrupt is also generated for errors recorded as Corrected error on reads.~~

When enabled, the fault handling interrupt is generated for corrected error events on reads.:

RCFI	Meaning
0b0	Fault handling interrupt not generated for corrected Corrected error events.errors on reads.
0b1	Fault handling interrupt generated for corrected Corrected error events.errors on reads.

If the node implements a corrected error counter or counters, and the counter or counters are enabled, then a corrected error event is defined as follows:

- A corrected error event occurs when a counter overflows and sets a counter overflow flag to 1.
- It is UNPREDICTABLE whether a corrected error event occurs when a software write sets a counter overflow flag to 1.
- It is UNPREDICTABLE whether a corrected error event occurs when a counter overflows and the overflow flag was previously set to 1.

Otherwise, a corrected error event occurs when the error record records an error as a Corrected error.

If the node implements a corrected error counter or counters and ERR<n>CTLR.CED is not implemented, then the counter or counters are always enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

WUE, bit [7]**When ERR<n>FR.UE == 0b11:**

In-band error response on writes enable.

When enabled, responses to writes that detect an error that is not corrected and is not deferred are signaled with an in-band error response (External Abort).

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester.

WUE	Meaning
0b0	In-band error response for uncorrected errors on writes disabled.
0b1	In-band error response for uncorrected errors on writes enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

WFI, bit [6]**When ERR<n>FR.FI == 0b11:**

Fault handling interrupt on writes enable.

When enabled:

- The fault handling interrupt is generated for errors recorded as either Deferred error or Uncorrected error on writes.
- If the corresponding fault handling interrupt for Corrected errors control is not implemented:
 - If the node implements Corrected error counters for writes, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1.
 - Otherwise, the fault handling interrupt is also generated for errors recorded as Corrected error on writes.

WFI	Meaning
0b0	Fault handling interrupt on writes disabled.
0b1	Fault handling interrupt on writes enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

WUI, bit [5]**When ERR<n>FR.UI == 0b11:**

Uncorrected error recovery interrupt on writes enable.

When enabled, the error recovery interrupt is generated for errors recorded as Uncorrected error on writes.

WUI	Meaning
0b0	Error recovery interrupt on writes disabled.
0b1	Error recovery interrupt on writes enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[4]**When ERR<n>FR.UE == 0b10:****UE, bit [0] of bit [4]**

In-band error response enable.

When [ERR<n>FR.UE](#) == 0b10, this control applies to errors arising from both reads and writes.

When enabled, responses to transactions that detect an error that is not corrected and is not deferred are signaled with an in-band error response (External Abort).

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester.

UE	Meaning
0b0	In-band error response for uncorrected errors disabled.
0b1	In-band error response for uncorrected errors enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When ERR<n>FR.UE == 0b11:**RUE, bit [0] of bit [4]**

In-band error response on reads enable.

When [ERR<n>FR.UE](#) == 0b11, this field is named RUE.

When enabled, responses to reads that detect an error that is not corrected and is not deferred are signaled with an in-band error response (External Abort).

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester.

RUE	Meaning
0b0	In-band error response for uncorrected errors on reads disabled.
0b1	In-band error response for uncorrected errors on reads enabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[3]

When ERR<n>FR.FI == 0b10:

FI, bit [0] of bit [3]

Fault handling interrupt enable.

When `ERR<n>FR.FI == 0b10`, this control applies to errors ~~on arising from~~ both reads and writes.

When enabled:

- The fault handling interrupt is generated for all errors recorded as ~~either Deferred error or~~ Uncorrected error.
- If the fault handling interrupt ~~control for DeferredCorrected errors,~~ `ERR<n>CTLR.DFI` ~~control~~ is not implemented, ~~then the fault handling interrupt is generated for all errors recorded as Deferred error.:~~
 - ~~If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1.~~
 - ~~Otherwise, the fault handling interrupt is also generated for all errors recorded as Corrected error.~~
- ~~If the fault handling interrupt controls for Deferred errors and corrected error events,~~ `ERR<n>CTLR.{DFI, CFI}`, are not implemented, then the fault handling interrupt is generated for all corrected error events.

FI	Meaning
0b0	Fault handling interrupt disabled.
0b1	Fault handling interrupt enabled.

See `ERR<n>CTLR.CFI` for more information on corrected error events.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When ERR<n>FR.FI == 0b11:

RFI, bit [0] of bit [3]

Fault handling interrupt on reads enable.

When `ERR<n>FR.FI == 0b11`, this field is named RFI.

When enabled:

- The fault handling interrupt is generated for errors recorded as ~~either Deferred error or~~ Uncorrected error on reads.

- If the corresponding fault handling interrupt control for DeferredCorrected errors, ERR<n>CTLR.RDFI, control is not implemented, then the fault handling interrupt is generated for errors recorded as Deferred error on reads.:
 - If the node implements Corrected error counters for reads, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1.
 - Otherwise, the fault handling interrupt is also generated for errors recorded as Corrected error on reads.
- If the corresponding fault handling interrupt controls for Deferred errors and corrected error events, ERR<n>CTLR.{RDFI, RCFI}, are not implemented, then the fault handling interrupt is generated for corrected error events on reads.

RFI	Meaning
0b0	Fault handling interrupt on reads disabled.
0b1	Fault handling interrupt on reads enabled.

See ERR<n>CTLR.CFI for more information on corrected error events.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit[2]

When ERR<n>FR.UI == 0b10:

UI, bit [0] of bit [2]

Uncorrected error recovery interrupt enable.

When ERR<n>FR.UI == 0b10, this control applies to errors arising from both reads and writes.

When enabled, the error recovery interrupt is generated for all errors recorded as Uncorrected error.

UI	Meaning
0b0	Error recovery interrupt disabled.
0b1	Error recovery interrupt enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

When ERR<n>FR.UI == 0b11:

RUI, bit [0] of bit [2]

Uncorrected error recovery interrupt on reads enable.

When ERR<n>FR.UI == 0b11, this field is named RUI.

When enabled, the error recovery interrupt is generated for errors recorded as Uncorrected error on reads.

RUI	Meaning
0b0	Error recovery interrupt on reads disabled.
0b1	Error recovery interrupt on reads enabled.

The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

IMPLEMENTATION DEFINED, bit [1]

Reserved for IMPLEMENTATION DEFINED controls. Must permit SBZP write policy for software.

ED, bit [0]

When ERR<n>FR.ED == 0b10:

Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Arm recommends that, when disabled, correct error detection and correction codes are written for writes, unless disabled by an IMPLEMENTATION DEFINED control for error injection.

ED	Meaning
0b0	Error reporting disabled.
0b1	Error reporting enabled.

It is IMPLEMENTATION DEFINED whether the node fully disables error detection and correction when reporting is disabled. That is, even with error reporting disabled, the node might continue to silently correct errors. Uncorrected errors might result in corrupt data being silently propagated by the node.

Note

If this node requires initialization after Cold reset to prevent signaling false errors, then Arm recommends this field is set to 0 on Cold reset, meaning errors are not reported from Cold reset. This allows boot software to initialize a node without signaling errors. Software can enable error reporting after the node is initialized. Otherwise, the Cold reset value is IMPLEMENTATION DEFINED. If the Cold reset value is 1, the reset values of other controls in this register are also IMPLEMENTATION DEFINED and should not be UNKNOWN.

The reset behavior of this field is:

- On an ErrorCold recovery reset, when RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an 0 IMPLEMENTATION DEFINED value.
- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to an IMPLEMENTATION DEFINED value.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to 0.

Otherwise:

Reserved, RES0.

Accessing ERR<n>CTRL

ERR<n>CTLR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
RAS	0x008 + (64 * n)	ERR<n>CTLR

Accesses to this interface are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

ERR<n>FR, Error Record <n> Feature Register, n = 0 - 65534

The ERR<n>FR characteristics are:

Purpose

Defines whether <n> is the first record owned by a node:

- If <n> is the first error record owned by a node, then ERR<n>FR.ED is not 0b00.
- If <n> is not the first error record owned by a node, then ERR<n>FR.ED is 0b00.

If <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configuration

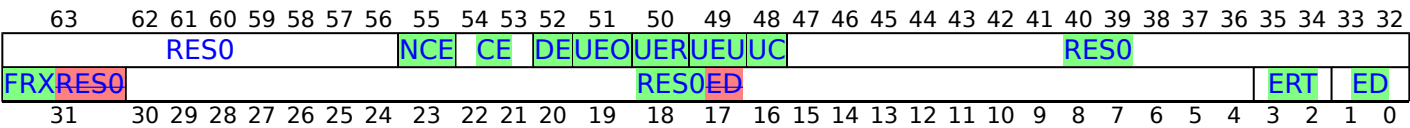
This register is present only when error record <n> is implemented. Otherwise, direct accesses to ERR<n>FR are RES0.

Attributes

ERR<n>FR is a 64-bit register.

Field descriptions

When error record <n> is not the first error record owned by the node:



Bits [63:562]

Reserved, RES0.

NCE, bit [55]

When RAS System Architecture v2 is implemented and ERR<q>FR.CEC != 0b000:

No countable errors. Describes whether this error record supports recording countable errors.

NCE	Meaning
0b0	Records countable errors.
0b1	Does not record countable errors.

When ERR<q>FR.CEC != 0b000, at least one error record owned by the node records countable errors.

Otherwise:

Reserved, RES0.

CE, bits [54:53]**When ERR<n>FR.FRX == 1:**

Corrected Error recording. Describes the types of Corrected errors the error record can record, if any.

CE	Meaning
0b00	Does not record Corrected errors.
0b01	Records only transient or persistent Corrected errors. That is, Corrected errors recorded by setting ERR<n>STATUS.CE to either 0b01 or 0b11.
0b10	Records only non-specific Corrected errors. That is, Corrected errors recorded by setting ERR<n>STATUS.CE to 0b10.
0b11	Records all types of Corrected error.

Otherwise:

Reserved, RES0.

DE, bit [52]**When ERR<n>FR.FRX == 1:**

Deferred Error recording. Describes whether the error record supports recording Deferred errors.

DE	Meaning
0b0	Does not record Deferred errors.
0b1	Records Deferred errors.

Otherwise:

Reserved, RES0.

UEO, bit [51]**When ERR<n>FR.FRX == 1:**

Latent or Restartable Error recording. Describes whether the error record supports recording Latent or Restartable errors.

UEO	Meaning
0b0	Does not record Latent or Restartable errors.
0b1	Records Latent or Restartable errors.

Otherwise:

Reserved, RES0.

UER, bit [50]**When ERR<n>FR.FRX == 1:**

Signaled or Recoverable Error recording. Describes whether the error record supports recording Signaled or Recoverable errors.

UER	Meaning
0b0	Does not record Signaled or Recoverable errors.
0b1	Records Signaled or Recoverable errors.

Otherwise:

Reserved, RES0.

UEU, bit [49]**When ERR<n>FR.FRX == 1:**

Unrecoverable Error recording. Describes whether the error record supports recording Unrecoverable errors.

UEU	Meaning
0b0	Does not record Unrecoverable errors.
0b1	Records Unrecoverable errors.

Otherwise:

Reserved, RES0.

UC, bit [48]**When ERR<n>FR.FRX == 1:**

Uncontainable Error recording. Describes whether the error record supports recording Uncontainable errors.

UC	Meaning
0b0	Does not record Uncontainable errors.
0b1	Records Uncontainable errors.

Otherwise:

Reserved, RES0.

Bits [47:32]

Reserved, RES0.

FRX, bit [31]**When Error record <n> is implemented, RAS System Architecture v2 is implemented and ERR<n>FR.ERT == 0b00:**

Feature Register extension. Defines whether ERR<n>FR[54:48] describe the error types supported by this error record.

FRX	Meaning
0b0	ERR<n>FR[54:48] are RES0.
0b1	ERR<n>FR[54:48] are defined by the architecture.

If ERR<n>FR.FRX is 0, and error record <n> is implemented, then the error types supported by this error record are as described by the first error record of this node.

Otherwise:

Reserved, RES0.

Bits [30:4]

Reserved, RES0.

ERT, bits [3:2]**When RAS System Architecture v2 is implemented:**

Error Record Type. Defines the type of error record.

ERT	Meaning
0b00	Error record <n> not implemented or is a normal record that is not the first error record of the node.
0b01	Error record <n> is a continuation record of the previous error record, <n-1>.

All other values are reserved.

Otherwise:

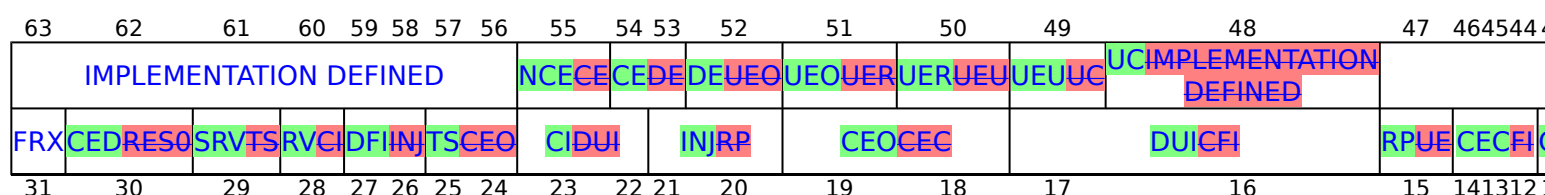
Reserved, RES0.

ED, bits [1:0]

Error reporting and logging. Indicates error record <n> is not the first error record owned the node.

ED	Meaning
0b00	Error record <n> is not implemented or is not the first error record owned by the node.

Access This to field this reads field is as RO0b00.

When error record <n> is the first error record owned by the node:**IMPLEMENTATION DEFINED, bits [63:56:55]****When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == != 01:**

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

NCE, bit [55]**When RAS System Architecture v2 is implemented and ERR<n>FR.CEC != 0b000:****NCE, bit [0] of bit [55:55]**

No countable errors. Describes whether this error record supports recording countable errors.

NCE	Meaning
0b0	Records countable errors.
0b1	Does not record countable errors.

When ERR<n>FR.CEC != 0b000, at least one error record owned by the node records countable errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0:**IMPLEMENTATION DEFINED, bit [0] of bit [55:55]**

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

CE, bits [54:53]**When ERR<n>FR.FRX == 1:**

Corrected Error recording. Describes the types of Corrected errors the node can record, if any.

CE	Meaning
0b00	Does not record Corrected errors.
0b01	Records only transient or persistent Corrected errors. That is, Corrected errors recorded by setting ERR<n>STATUS.CE to either 0b01 or 0b11.
0b10	Records only non-specific Corrected errors. That is, Corrected errors recorded by setting ERR<n>STATUS.CE to 0b10.
0b11	Records all types of Corrected error.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

DE, bit [52]**When ERR<n>FR.FRX == 1:**

Deferred Error recording. Describes whether the node supports recording Deferred errors.

DE	Meaning
0b0	Does not record Deferred errors.
0b1	Records Deferred errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

UEO, bit [51]**When ERR<n>FR.FRX == 1:**

Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors.

UEO	Meaning
0b0	Does not record Latent or Restartable errors.
0b1	Records Latent or Restartable errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0 Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

UER, bit [50]**When ERR<n>FR.FRX == 1:**

Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors.

UER	Meaning
0b0	Does not record Signaled or Recoverable errors.
0b1	Records Signaled or Recoverable errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0 Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

UEU, bit [49]**When ERR<n>FR.FRX == 1:**

Unrecoverable Error recording. Describes whether the node supports recording Unrecoverable errors.

UEU	Meaning
0b0	Does not record Unrecoverable errors.
0b1	Records Unrecoverable errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0 Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

UC, bit [48]**When ERR<n>FR.FRX == 1:**

Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.

UC	Meaning
0b0	Does not record Uncontainable errors.
0b1	Records Uncontainable errors.

When RAS System Architecture v2 is not implemented and ERR<n>FR.FRX == 0 Otherwise:

Reserved for identifying IMPLEMENTATION DEFINED controls.

Otherwise:

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [47:32]

Reserved for identifying IMPLEMENTATION DEFINED controls.

FRX, bit [31]**When RAS System Architecture v1.1 is implemented:**

Feature Register extension. Defines whether ERR<n>FR[63:48] are architecturally defined.

FRX	Meaning
0b0	ERR<n>FR[63:48] are IMPLEMENTATION DEFINED.
0b1	ERR<n>FR[63:48] are defined by the architecture.

Otherwise:

Reserved, RES0.

CED, Bits bit [30:26]**When RAS System Architecture v2 is implemented and ERR<n>FR.CEC != 0b000:**

Error counter disable. Indicates whether the node implements a control to disable any implemented Corrected error counters.

CED	Meaning
0b0	Error counter disable control is not implemented and the error counter(s) are always enabled. ERR<n>CTLR.CED is RES0.
0b1	Enabling and disabling of error counter(s) is supported and controlled by ERR<n>CTLR.CED.

Otherwise:

Reserved, RES0.

SRV, bit [29]**When RAS System Architecture v2 is implemented:**

Status Reset Value. Indicates how node <n> and each error record <m> owned by node <n> is reset.

SRV	Meaning
0b0	Node <n> and each error record <m> owned by node <n> are reset as follows: <ul style="list-style-type: none"> ERR<m>STATUS.{AV, V, MV} are set to {0, 0, 0} on a Cold reset and preserved on Error Recovery reset. ERR<n>CTLR.ED is set to an IMPLEMENTATION DEFINED value on a Cold reset and preserved on Error Recovery reset.
0b1	Node <n> and each error record <m> owned by node <n> are reset as follows: <ul style="list-style-type: none"> ERR<m>STATUS.{AV, V, MV} are set to architecturally UNKNOWN values on a Cold reset and preserved on Error Recovery reset. ERR<n>CTLR.ED is set to 0 on both Cold reset and Error Recovery reset.

All other values are reserved.

Otherwise:

Reserved, RES0.

RV, bit [28]

When RAS System Architecture v2 is implemented:

Reset Valid. Indicates whether each error record <m> implemented by the node includes the Reset Valid flag, ERR<m>STATUS.RV.

RV	Meaning
0b0	ERR<m>STATUS.RV is RES0.
0b1	ERR<m>STATUS.RV is a R/W1C bit set to 1 on Error Recovery reset.

All other values are reserved.

Otherwise:

Reserved, RES0.

DFI, bits [27:26]

When RAS System Architecture v2 is implemented and ERR<n>FR.FI != 0b00:

Fault handling interrupt for deferred errors control. Indicates whether the enabling and disabling of fault handling interrupts on deferred errors is supported by the node.

DFI	Meaning
0b00	Does not support the enabling and disabling of fault handling interrupts on deferred errors. ERR<n>CTLR.DFI is RES0.
0b10	Enabling and disabling of fault handling interrupts on deferred errors is supported and controllable using ERR<n>CTLR.DFI.
0b11	Enabling and disabling of fault handling interrupts on deferred errors is supported, and controllable using ERR<n>CTLR.WDFI for writes and ERR<n>CTLR.RDFI for reads.

All other values are reserved.

Otherwise:

Reserved, RES0.

TS, bits [25:24]

Timestamp Extension. Indicates whether, for each error record <m> owned by this node, [ERR<m>MISC3](#) is used as the timestamp register, and, if it is, the timebase used by the timestamp.

TS	Meaning
0b00	Does not support a timestamp register.
0b01	Implements a timestamp register in ERR<n>MISC3 for each error record <m> owned by the node. The timestamp uses the same timebase as the system Generic Timer.
Note For an error record that has an affinity to a PE, this is the same timer that is visible through CNTPCT_ELO at the highest Exception level on that PE.	
0b10	Implements a timestamp register in ERR<m>MISC3 for each error record <m> owned by the node. The timestamp uses an IMPLEMENTATION DEFINED timebase.

All other values are reserved.

CI, bits [23:22]

Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented by the node.

CI	Meaning
0b00	Does not support the critical error interrupt. ERR<n>CTLR.CI is RES0.
0b01	Critical error interrupt is supported and always enabled. ERR<n>CTLR.CI is RES0.
0b10	Critical error interrupt is supported and controllable using ERR<n>CTLR.CI .

All other values are reserved.

INJ, bits [21:20]

Fault Injection Extension. Indicates whether the Common Fault Injection Model Extension is implemented by the node.

INJ	Meaning
0b00	Does not support the Common Fault Injection Model Extension.
0b01	Supports the Common Fault Injection Model Extension. See ERR<n>PFGF for more information.

All other values are reserved.

CEO, bits [19:18]**When ERR<n>FR.CEC != 0b000:**

Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node.

CEO	Meaning
0b00	Keeps the previous error syndrome.
0b01	If ERR<m>STATUS.OF is 1 before the Corrected error is counted, then the error record keeps the previous syndrome. Otherwise the previous syndrome is overwritten.

All other values are reserved.

The second or subsequent Corrected error is counted by the Corrected error counter, regardless of the value of this field. If counting the error causes unsigned overflow of the counter, then [ERR<m>STATUS.OF](#) is set to 1.

This means that, if no other error is subsequently recorded that overwrites the syndrome:

- If ERR<n>FR.CEO is 0b00, the error record holds the syndrome for the first recorded Corrected error.
- If ERR<n>FR.CEO is 0b01, the error record holds the syndrome for the most recently recorded Corrected error before the counter overflows.

Otherwise:

Reserved, RES0.

DUI, bits [17:16]

When ERR<n>FR.UI != 0b00:

Error recovery interrupt for deferred errors control. Indicates whether the enabling and disabling of error recovery interrupts on deferred errors is supported by the node.

DUI	Meaning
0b00	Does not support the enabling and disabling of error recovery interrupts on deferred errors. ERR<n>CTLR.DUI is RES0.
0b10	Enabling and disabling of error recovery interrupts on deferred errors is supported and controllable using ERR<n>CTLR.DUI .
0b11	Enabling and disabling of error recovery interrupts on deferred errors is supported, and controllable using ERR<n>CTLR.WDUI for writes and ERR<n>CTLR.RDUI for reads.

All other values are reserved.

Otherwise:

Reserved, RES0.

RP, bit [15]

When ERR<n>FR.CEC != 0b000:

Repeat counter. Indicates whether the node implements a second Corrected error counter in [ERR<m>MISC0](#) for each error record <m> owned by the node that can record countable errors.

RP	Meaning
0b0	Implements a single Corrected error counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors.
0b1	Implements a first (repeat) counter and a second (other) counter in ERR<m>MISC0 for each error record <m> owned by the node that can record countable errors. The repeat counter is the same size as the primary error counter.

Otherwise:

Reserved, RES0.

CEC, bits [14:12]

Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter mechanisms in [ERR<m>MISC0](#) for each error record <m> owned by the node that can record countable errors.

CEC	Meaning
0b000	Does not implement the standard Corrected error counter model.
0b010	Implements an 8-bit Corrected error counter in ERR<m>MISC0 [39:32] for each error record <m> owned by the node that can record countable errors.
0b100	Implements a 16-bit Corrected error counter in ERR<m>MISC0 [47:32] for each error record <m> owned by the node that can record countable errors.

All other values are reserved.

Note

Implementations might include other error counter models, or might include the standard model and not indicate this in ERR<n>FR.

CFI, bits [11:10]

When ERR<n>FR.FI != 0b00:

Fault handling interrupt for corrected errors control. Indicates whether the enabling and disabling of fault handling interrupts on corrected errors is supported by the node.

CFI	Meaning
0b00	Does not support the enabling and disabling of fault handling interrupts on corrected errors. ERR<n>CTLR .CFI is RES0.
0b10	Enabling and disabling of fault handling interrupts on corrected errors is supported and controllable using ERR<n>CTLR .CFI.
0b11	Enabling and disabling of fault handling interrupts on corrected errors is supported, and controllable using ERR<n>CTLR .WCFI for writes and ERR<n>CTLR .RCFI for reads.

All other values are reserved.

Otherwise:

Reserved, RES0.

UE, bits [9:8]

In-band error response (External Abort). Indicates whether the in-band error response and associated controls are implemented by the node.

UE	Meaning
0b00	Does not support the in-band error response. ERR<n>CTLR .UE is RES0.
0b01	In-band error response is supported and always enabled. ERR<n>CTLR .UE is RES0.
0b10	In-band error response is supported and controllable using ERR<n>CTLR .UE.
0b11	In-band error response is supported, and controllable using ERR<n>CTLR .WUE for writes and ERR<n>CTLR .RUE for reads.

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester.

FI, bits [7:6]

Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented by the node.

FI	Meaning
0b00	Does not support the fault handling interrupt. ERR<n>CTLR.FI is RES0.
0b01	Fault handling interrupt is supported and always enabled. ERR<n>CTLR.FI is RES0.
0b10	Fault handling interrupt is supported and controllable using ERR<n>CTLR.FI .
0b11	Fault handling interrupt is supported, and controllable using ERR<n>CTLR.WFI for writes and ERR<n>CTLR.RFI for reads.

UI, bits [5:4]

Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented by the node.

UI	Meaning
0b00	Does not support the error handling interrupt. ERR<n>CTLR.UI is RES0.
0b01	Error handling interrupt is supported and always enabled. ERR<n>CTLR.UI is RES0.
0b10	Error handling interrupt is supported and controllable using ERR<n>CTLR.UI .
0b11	Error handling interrupt is supported, and controllable using ERR<n>CTLR.WUI for writes and ERR<n>CTLR.RUI for reads.

IMPLEMENTATION DEFINED, bits [3:2]

IMPLEMENTATION DEFINED.

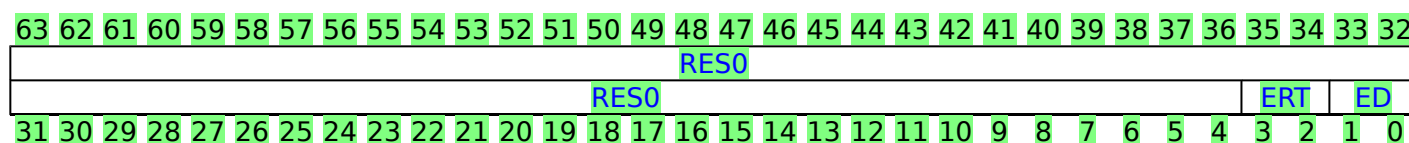
ED, bits [1:0]

Error reporting and logging. Indicates error record <n> is **a normal record and** the first record owned the node, and whether the node implements the controls for enabling and disabling error reporting and logging.

ED	Meaning
0b01	Error reporting and logging always enabled. ERR<n>CTLR.ED is RES0.
0b10	Error reporting and logging is controllable using ERR<n>CTLR.ED .

All other values are reserved.

When RAS System Architecture v2 is implemented and error record <n> is a proxy for a RAS agent:

**Bits [63:4]**

Reserved, RES0.

ERT, bits [3:2]

Error Record Type. Defines the type of error record.

ERT	Meaning
0b01	Error record is a proxy for a RAS agent.

All other values are reserved.

Access to this field is **RO**.

ED, bits [1:0]

Error reporting and logging. Indicates error record <n> is not a true error record.

ED	Meaning
0b11	Error record <n> is not an error record.

Access to this field is **RO**.

Accessing ERR<n>FR

ERR<n>FR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
RAS	0x000 + (64 * n)	ERR<n>FR

Accesses to this interface are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ERR<n>STATUS, Error Record <n> Primary Status Register, n = 0 - 65534

The ERR<n>STATUS characteristics are:

Purpose

Contains status information for error record <n>, including:

- Whether any error has been detected (valid).
- Whether any detected error was not corrected, and returned to a Requester.
- Whether any detected error was not corrected and deferred.
- Whether an error record has been discarded because additional errors have been detected before the first error was handled by software (overflow).
- Whether any error has been reported.
- Whether the other error record registers contain valid information.
- Whether the error was reported because poison data was detected or because a corrupt value was detected by an error detection code.
- A primary error code.
- An IMPLEMENTATION DEFINED extended error code.

Within this register:

- ERR<n>STATUS.{AV, V, MV} are valid bits that define whether error record <n> registers are valid.
- ERR<n>STATUS.{UE, OF, CE, DE, UET} encode the types of error or errors recorded.
- ERR<n>STATUS.{CI, ER, PN, IERR, SERR} are syndrome fields.

Configuration

This register is present only when error record <n> is implemented. Otherwise, direct accesses to ERR<n>STATUS are RES0.

[ERR<q>FR](#) describes the features implemented by the node that owns error record <n>. <q> is the index of the first error record owned by the same node as error record <n>. If the node owns a single record, then q = n.

For IMPLEMENTATION DEFINED fields in ERR<n>STATUS, writing zero returns the error record to an initial quiescent state.

In particular, if any IMPLEMENTATION DEFINED syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, non-zero, and ignore writes are compliant with this requirement.

Note

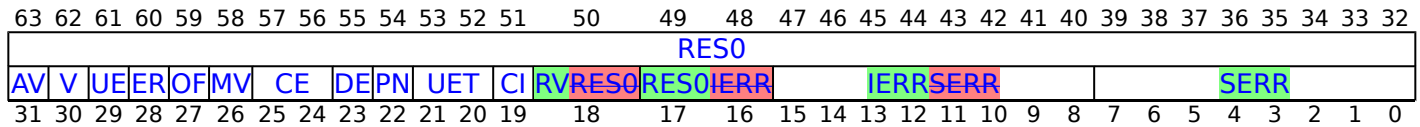
Arm recommends that any IMPLEMENTATION DEFINED syndrome field that can generate a Fault Handling, Error Recovery, Critical, or IMPLEMENTATION DEFINED, interrupt request is disabled at Cold reset and is enabled by software writing an IMPLEMENTATION DEFINED nonzero value to an IMPLEMENTATION DEFINED field in [ERR<q>CTLR](#).

Attributes

ERR<n>STATUS is a 64-bit register.

Field descriptions

When RAS System Architecture v1.1 is implemented:



Bits [63:32]

Reserved, RES0.

AV, bit [31]

When error record <n> includes an address associated with an error:

Address Valid.

AV	Meaning
0b0	ERR<n>ADDR not valid.
0b1	ERR<n>ADDR contains an address associated with the highest priority error recorded by this record.

The reset behavior of this field is:

- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- On a Cold reset, this field resets to 0.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

V, bit [30]

Status Register Valid.

V	Meaning
0b0	ERR<n>STATUS not valid.
0b1	ERR<n>STATUS valid. At least one error has been recorded.

The reset behavior of this field is:

- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- On a Cold reset, this field resets to 0.

Access to this field is **W1C**.

UE, bit [29]

Uncorrected Error.

UE	Meaning
0b0	No errors have been detected, or all detected errors have been either corrected or deferred.
0b1	At least one detected error was not corrected and not deferred.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

ER, bit [28]

Error Reported.

ER	Meaning
0b0	No in-band error response (External Abort) signaled to the Requester making the access or other transaction.
0b1	An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true: <ul style="list-style-type: none"> The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is implemented and was 1 when an error was detected and not corrected. The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is not implemented and the component always reports errors.

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as a Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester, causing this field to be set to 1. If no in-band error response to the Requester, this field is set to 0.

Note

An in-band error response signaled by the component might be masked and not generate any exception.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE == 0
 - this field can be set to 0b1 by a Deferred error
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - this field is never set to 0b1 by a Deferred error
- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

OF, bit [27]

Overflow.

Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs:

- A Corrected error counter is implemented, an error is counted, and the counter overflows.
- ERR<n>STATUS.V was previously 1, a Corrected error counter is not implemented, and a Corrected error is recorded.
- ERR<n>STATUS.V was previously 1, and a type of error other than a Corrected error is recorded.

Otherwise, this field is unchanged when an error is recorded.

If a Corrected error counter is implemented:

- A direct write that modifies the counter overflow flag indirectly might set this field to an UNKNOWN value.
- A direct write to this field that clears this field to zero might indirectly set the counter overflow flag to an UNKNOWN value.

OF	Meaning
0b0	Since this field was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed.
0b1	Since this field was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

MV, bit [26]

When error record <n> includes an additional information for an error:

Miscellaneous Registers Valid.

MV	Meaning
0b0	ERR<n>MISC<m> not valid.
0b1	The contents of the ERR<n>MISC<m> registers contain additional information for an error recorded by this record.

Note

If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.

The reset behavior of this field is:

- **On a Cold reset:**
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- **On a Cold reset, this field resets to 0.**

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

CE, bits [25:24]

Corrected Error.

CE	Meaning
0b00	No errors were corrected.
0b01	At least one transient error was corrected.
0b10	At least one error was corrected.
0b11	At least one persistent error was corrected.

The mechanism by which a component or node detects whether a Corrected error is transient or persistent is IMPLEMENTATION DEFINED. If no such mechanism is implemented, then the node sets this field to 0b10 when a corrected error is recorded.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

DE, bit [23]

Deferred Error.

DE	Meaning
0b0	No errors were deferred.
0b1	At least one error was not corrected and deferred.

Support for deferring errors is IMPLEMENTATION DEFINED.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

PN, bit [22]

Poison.

PN	Meaning
0b0	Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded.
0b1	Uncorrected error or Deferred error recorded because a poison value was detected.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if any of the following are true:
 - ERR<n>STATUS.V == 0
 - ERR<n>STATUS.DE == 0 and ERR<n>STATUS.UE == 0
- Otherwise, access to this field is **W1C**.

UET, bits [21:20]

Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error.

UET	Meaning
0b00	Uncorrected error, Uncontainable error (UC).
0b01	Uncorrected error, Unrecoverable error (UEU).
0b10	Uncorrected error, Latent or Restartable error (UEO).
0b11	Uncorrected error, Signaled or Recoverable error (UER).

Note

Software might use the information in the error record registers to determine what recovery is necessary.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if any of the following are true:
 - ERR<n>STATUS.V == 0
 - ERR<n>STATUS.UE == 0
- Otherwise, access to this field is **W1C**.

CI, bit [19]

Critical Error. Indicates whether a critical error condition has been recorded.

CI	Meaning
0b0	No critical error condition.
0b1	Critical error condition.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

RV, Bits bit [18:16]**When RAS System Architecture v2 is implemented:**

Reset Valid. When ERR<n>STATUS.V is 1, indicating the error record is valid, this field indicates whether the error was recorded before or after the most recent Error Recovery reset.

RV	Meaning
0b0	If the error record is valid then it was recorded after the last Error Recovery reset.
0b1	If the error record is valid then it was recorded before the last Error Recovery reset.

This field is set to 0 when an error is recorded.

The reset behavior of this field is:

- On an Error recovery reset, this field resets to 1.

Access to this field is **W1C**.

Otherwise:

Reserved, RES0.

Bits [17:16]

Reserved, RES0.

IERR, bits [15:8]

IMPLEMENTATION DEFINED error code. Used with any primary error code ERR<n>STATUS.SERR value. Further IMPLEMENTATION DEFINED information can be placed in the ERR<n>MISC<m> registers.

The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN.

Note

This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - the Common Fault Injection Model Extension is not implemented by the node that owns this error record
 - ERR<n>STATUS.V == 0
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<q>PFGF.SYN == 0
 - ERR<n>STATUS.V == 0
- Otherwise, access to this field is **RW**.

SERR, bits [7:0]

Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry.

SERR	Meaning
0x00	No error.
0x01	IMPLEMENTATION DEFINED error.
0x02	Data value from (non-associative) internal memory. For example, ECC from on-chip SRAM or buffer.
0x03	IMPLEMENTATION DEFINED pin. For example, nSEI pin.
0x04	Assertion failure. For example, consistency failure.
0x05	Error detected on internal data path. For example, parity on ALU result.
0x06	Data value from associative memory. For example, ECC error on cache data.
0x07	Address/control value from associative memory. For example, ECC error on cache tag.
0x08	Data value from a TLB. For example, ECC error on TLB data.
0x09	Address/control value from a TLB. For example, ECC error on TLB tag.
0x0A	Data value from producer. For example, parity error on write data bus.
0x0B	Address/control value from producer. For example, parity error on address bus.
0x0C	Data value from (non-associative) external memory. For example, ECC error in SDRAM.
0x0D	Illegal address (software fault). For example, access to unpopulated memory.
0x0E	Illegal access (software fault). For example, byte write to word register.
0x0F	Illegal state (software fault). For example, device not ready.
0x10	Internal data register. For example, parity on a SIMD&FP register. For a PE, all general-purpose, stack pointer, SIMD&FP, and SVE registers are data registers.
0x11	Internal control register. For example, Parity on a System register. For a PE, all registers other than general-purpose, stack pointer, SIMD&FP, and SVE registers are control registers.
0x12	Error response from Completer of access. For example, error response from cache write-back.
0x13	External timeout. For example, timeout on interaction with another component.
0x14	Internal timeout. For example, timeout on interface within the component.
0x15	Deferred error from Completer not supported at Requester. For example, poisoned data received from the Completer of an access by a Requester that cannot defer the error further.
0x16	Deferred error from Requester not supported at Completer. For example, poisoned data received from the Requester of an access by a Completer that cannot defer the error further.
0x17	Deferred error from Completer passed through. For example, poisoned data received from the Completer of an access and returned to the Requester.
0x18	Deferred error from Requester passed through. For example, poisoned data received from the Requester of an access and deferred to the Completer.
0x19	Error recorded by PCIe error logs. Indicates that the component has recorded an error in a PCIe error log. This might be the PCIe device status register, AER, DVSEC, or other mechanisms defined by PCIe.
0x1A	Other internal error. For example, parity error on internal state of the component that is not covered by another primary error code.

All other values are reserved.

The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN.

Note

This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - the Common Fault Injection Model Extension is not implemented by the node that owns this error record
 - ERR<n>STATUS.V == 0
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<q>PFGF.SYN == 0
 - ERR<n>STATUS.V == 0
- Otherwise, access to this field is **RW**.

When RAS System Architecture v1.0 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
AV	V	UE	ER	OF	MV	CE	DE	PN	UET	RES0						IERR						SERR									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

AV, bit [31]

When error record <n> includes an address associated with an error:

Address Valid.

AV	Meaning
0b0	ERR<n>ADDR not valid.
0b1	ERR<n>ADDR contains an address associated with the highest priority error recorded by this record.

The reset behavior of this field is:

- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- On a Cold reset, this field resets to 0.

Accessing this field has the following behavior:

- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

Otherwise:

Reserved, RES0.

V, bit [30]

Status Register Valid.

V	Meaning
0b0	ERR<n>STATUS not valid.
0b1	ERR<n>STATUS valid. At least one error has been recorded.

The reset behavior of this field is:

- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- On a Cold reset, this field resets to 0.

Accessing this field has the following behavior:

- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

UE, bit [29]

Uncorrected Error.

UE	Meaning
0b0	No errors have been detected, or all detected errors have been either corrected or deferred.
0b1	At least one detected error was not corrected and not deferred.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.OF == 1
 - ERR<n>STATUS.OF is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

ER, bit [28]

Error Reported.

ER	Meaning
0b0	No in-band error response (External Abort) signaled to the Requester making the access or other transaction.
0b1	An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true: <ul style="list-style-type: none"> The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is implemented and was 1 when an error was detected and not corrected. The applicable one of the ERR<q>CTLR.{WUE, RUE, UE} fields is not implemented and the component always reports errors.

It is IMPLEMENTATION DEFINED whether an uncorrected error that is deferred and recorded as a Deferred error, but is not deferred to the Requester, will signal an in-band error response to the Requester, causing this field to be set to 1. If no in-band error response to the Requester, this field is set to 0.

Note

An in-band error response signaled by the component might be masked and not generate any exception.

If this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero, when any of:

- Clearing ERR<n>STATUS.V to 0.
- Clearing ERR<n>STATUS.UE to 0, if this field is never set to 1 by a Deferred error.
- Clearing ERR<n>STATUS.{UE,DE} to {0,0}, if this field can be set to 1 by a Deferred error.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE == 0
 - this field can be set to 0b1 by a Deferred error
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - this field is never set to 0b1 by a Deferred error
- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

OF, bit [27]

Overflow.

Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs:

- An Uncorrected error is detected and ERR<n>STATUS.UE == 1.
- A Deferred error is detected, ERR<n>STATUS.UE == 0 and ERR<n>STATUS.DE == 1.
- A Corrected error is detected, no Corrected error counter is implemented, ERR<n>STATUS.UE == 0, ERR<n>STATUS.DE == 0, and ERR<n>STATUS.CE != 0b00. ERR<n>STATUS.CE might be updated for the new Corrected error.

- A Corrected error counter is implemented, ERR<n>STATUS.UE == 0, ERR<n>STATUS.DE == 0, and the counter overflows.

It is IMPLEMENTATION DEFINED whether this field is set to 1 when one of the following occurs:

- A Deferred error is detected and ERR<n>STATUS.UE == 1.
- A Corrected error is detected, no Corrected error counter is implemented, and ERR<n>STATUS.{UE, DE} != {0, 0}.
- A Corrected error counter is implemented, ERR<n>STATUS.{UE, DE} != {0, 0}, and the counter overflows.

It is IMPLEMENTATION DEFINED whether this field is cleared to 0 when one of the following occurs:

- An Uncorrected error is detected and ERR<n>STATUS.UE == 0.
- A Deferred error is detected, ERR<n>STATUS.UE == 0, and ERR<n>STATUS.DE == 0.
- A Corrected error is detected, ERR<n>STATUS.UE == 0, ERR<n>STATUS.DE == 0, and ERR<n>STATUS.CE == 0b00.

The IMPLEMENTATION DEFINED clearing of this field might also depend on the value of the other error status fields.

If a Corrected error counter is implemented:

- A direct write that modifies the counter overflow flag indirectly might set this field to an UNKNOWN value.
- A direct write to this field that clears this field to 0 might indirectly set the counter overflow flag to an UNKNOWN value.

OF	Meaning
0b0	<p>If ERR<n>STATUS.UE == 1, then no error syndrome for an Uncorrected error has been discarded.</p> <p>If ERR<n>STATUS.UE == 0 and ERR<n>STATUS.DE == 1, then no error syndrome for a Deferred error has been discarded.</p> <p>If ERR<n>STATUS.UE == 0, ERR<n>STATUS.DE == 0, and a Corrected error counter is implemented, then the counter has not overflowed.</p> <p>If ERR<n>STATUS.UE == 0, ERR<n>STATUS.DE == 0, ERR<n>STATUS.CE != 0b00, and no Corrected error counter is implemented, then no error syndrome for a Corrected error has been discarded.</p> <hr/> <p>Note</p> <p>This field might have been set to 1 when an error syndrome was discarded and later cleared to 0 when a higher priority syndrome was recorded.</p> <hr/>
0b1	At least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **W1C**.

MV, bit [26]

When error record <n> includes an additional information for an error:

Miscellaneous Registers Valid.

MV	Meaning
0b0	ERR<n>MISC<m> not valid.
0b1	The contents of the ERR<n>MISC<m> registers contain additional information for an error recorded by this record. The IMPLEMENTATION DEFINED contents of the ERR<n>MISC<m> registers contains additional information for an error recorded by this record.

Note

If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.

The reset behavior of this field is:

- On a Cold reset:
 - When RAS System Architecture v2 is not implemented or ERR<n>FR.SRV == 0, this field resets to 0.
 - When RAS System Architecture v2 is implemented and ERR<n>FR.SRV == 1, this field resets to an architecturally UNKNOWN value.
- On a Cold reset, this field resets to 0.

Accessing this field has the following behavior:

- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

Otherwise:

Reserved, RES0.

CE, bits [25:24]

Corrected Error.

CE	Meaning
0b00	No errors were corrected.
0b01	At least one transient error was corrected.
0b10	At least one error was corrected.
0b11	At least one persistent error was corrected.

The mechanism by which a component or node detects whether a Corrected error is transient or persistent is IMPLEMENTATION DEFINED. If no such mechanism is implemented, then the node sets this field to 0b10 when a corrected error is recorded.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.OF == 1
 - ERR<n>STATUS.OF is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

DE, bit [23]

Deferred Error.

DE	Meaning
0b0	No errors were deferred.
0b1	At least one error was not corrected and deferred.

Support for deferring errors is IMPLEMENTATION DEFINED.

When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When ERR<n>STATUS.V == 0, access to this field is **UNKNOWN/WI**.
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.OF == 1
 - ERR<n>STATUS.OF is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

PN, bit [22]

Poison.

PN	Meaning
0b0	Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded.
0b1	Uncorrected error or Deferred error recorded because a poison value was detected.

If this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero, when any of:

- Clearing ERR<n>STATUS.V to 0.
- Clearing both ERR<n>STATUS.{DE, UE} to 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if any of the following are true:
 - ERR<n>STATUS.V == 0
 - ERR<n>STATUS.DE == 0 and ERR<n>STATUS.UE == 0
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:

- ERR<n>STATUS.UE != 0
- ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

UET, bits [21:20]

Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error.

UET	Meaning
0b00	Uncorrected error, Uncontainable error (UC).
0b01	Uncorrected error, Unrecoverable error (UEU).
0b10	Uncorrected error, Latent or Restartable error (UEO).
0b11	Uncorrected error, Signaled or Recoverable error (UER).

Note

Software might use the information in the error record registers to determine what recovery is necessary.

If this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero, when any of:

- Clearing ERR<n>STATUS.V to 0.
- Clearing ERR<n>STATUS.UE to 0.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if any of the following are true:
 - ERR<n>STATUS.V == 0
 - ERR<n>STATUS.UE == 0
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **W1C**.

Bits [19:16]

Reserved, RES0.

IERR, bits [15:8]

IMPLEMENTATION DEFINED error code. Used with any primary error code ERR<n>STATUS.SERR value. Further IMPLEMENTATION DEFINED information can be placed in the ERR<n>MISC<m> registers.

The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN.

Note

This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - the Common Fault Injection Model Extension is not implemented by the node that owns this error record
 - ERR<n>STATUS.V == 0
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<q>PFGF.SYN == 0
 - ERR<n>STATUS.V == 0
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **RW**.

SERR, bits [7:0]

Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry.

SERR	Meaning
0x00	No error.
0x01	IMPLEMENTATION DEFINED error.
0x02	Data value from (non-associative) internal memory. For example, ECC from on-chip SRAM or buffer.
0x03	IMPLEMENTATION DEFINED pin. For example, nSEI pin.
0x04	Assertion failure. For example, consistency failure.
0x05	Error detected on internal data path. For example, parity on ALU result.
0x06	Data value from associative memory. For example, ECC error on cache data.
0x07	Address/control value from associative memory. For example, ECC error on cache tag.
0x08	Data value from a TLB. For example, ECC error on TLB data.
0x09	Address/control value from a TLB. For example, ECC error on TLB tag.
0x0A	Data value from producer. For example, parity error on write data bus.
0x0B	Address/control value from producer. For example, parity error on address bus.
0x0C	Data value from (non-associative) external memory. For example, ECC error in SDRAM.
0x0D	Illegal address (software fault). For example, access to unpopulated memory.
0x0E	Illegal access (software fault). For example, byte write to word register.
0x0F	Illegal state (software fault). For example, device not ready.
0x10	Internal data register. For example, parity on a SIMD&FP register. For a PE, all general-purpose, stack pointer, SIMD&FP, and SVE registers are data registers.
0x11	Internal control register. For example, Parity on a System register. For a PE, all registers other than general-purpose, stack pointer, SIMD&FP, and SVE registers are control registers.
0x12	Error response from Completer of access. For example, error response from cache write-back.
0x13	External timeout. For example, timeout on interaction with another component.
0x14	Internal timeout. For example, timeout on interface within the component.
0x15	Deferred error from Completer not supported at Requester. For example, poisoned data received from the Completer of an access by a Requester that cannot defer the error further.
0x16	Deferred error from Requester not supported at Completer. For example, poisoned data received from the Requester of an access by a Completer that cannot defer the error further.
0x17	Deferred error from Completer passed through. For example, poisoned data received from the Completer of an access and returned to the Requester.
0x18	Deferred error from Requester passed through. For example, poisoned data received from the Requester of an access and deferred to the Completer.
0x19	Error recorded by PCIe error logs. Indicates that the component has recorded an error in a PCIe error log. This might be the PCIe device status register, AER, DVSEC, or other mechanisms defined by PCIe.
0x1A	Other internal error. For example, parity error on internal state of the component that is not covered by another primary error code.

All other values are reserved.

The implemented set of valid values that this field can take is IMPLEMENTATION DEFINED. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN.

Note

This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **UNKNOWN/WI** if all of the following are true:
 - the Common Fault Injection Model Extension is not implemented by the node that owns this error record
 - ERR<n>STATUS.V == 0
- Access is **UNKNOWN/WI** if all of the following are true:
 - ERR<q>PFGF.SYN == 0
 - ERR<n>STATUS.V == 0
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.DE == 0
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.CE != 0b00
 - ERR<n>STATUS.CE is not being cleared to 0b00 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE == 0
 - ERR<n>STATUS.DE != 0
 - ERR<n>STATUS.DE is not being cleared to 0b0 in the same write
- Access is **RO** if all of the following are true:
 - ERR<n>STATUS.UE != 0
 - ERR<n>STATUS.UE is not being cleared to 0b0 in the same write
- Otherwise, access to this field is **RW**.

Accessing ERR<n>STATUS

ERR<n>STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is IMPLEMENTATION DEFINED. See also [ERR<n>PFGF.SYN](#).

After reading ERR<n>STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

When RAS System Architecture v1.0 is implemented:

- Writes to ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{ER, PN, UET, IERR, SERR} syndrome fields are ignored if the highest priority nonzero error status field is not being cleared to zero. The error status fields in priority order from highest to lowest, are ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as UNKNOWN where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS allow a node to implement such a field as a fixed read-only value.

For example, when RAS System Architecture v1.1 is implemented, a write to ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as UNKNOWN values when ERR<n>STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- Write ones to all the W1C fields that are nonzero in the read value.

An exception is when the node supports writing to these fields as part of fault injection. See also [ERR<n>PFGF.SYN](#).

Component	Offset	Instance
RAS	0x010 + (64 * n)	ERR<n>STATUS

- When $ERR<n>STATUS.V \neq 0$, $ERR<n>STATUS.V$ is not being cleared to 0b0 in the same write and RAS System Architecture v1.1 is implemented, accesses to this register are **RO**.
- When $ERR<n>STATUS.UE \neq 0$, $ERR<n>STATUS.UE$ is not being cleared to 0b0 in the same write and RAS System Architecture v1.1 is implemented, accesses to this register are **RO**.
- When $ERR<n>STATUS.OF \neq 0$, $ERR<n>STATUS.OF$ is not being cleared to 0b0 in the same write and RAS System Architecture v1.1 is implemented, accesses to this register are **RO**.
- When $ERR<n>STATUS.CE \neq 0b00$, $ERR<n>STATUS.CE$ is not being cleared to 0b00 in the same write and RAS System Architecture v1.1 is implemented, accesses to this register are **RO**.
- When $ERR<n>STATUS.DE \neq 0$, $ERR<n>STATUS.DE$ is not being cleared to 0b0 in the same write and RAS System Architecture v1.1 is implemented, accesses to this register are **RO**.
- Otherwise, accesses to this register are **RW**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

The GICD_INMIR<n>E characteristics are:

Holds whether the corresponding SPI in the extended SPI range has the non-maskable property.

This register is present only when FEAT_GICv3p1 is implemented and FEAT_GICv3_NMI is implemented. Otherwise, direct accesses to GICD_INMIR<n>E are RES0.

When [GICD_TYPER](#).ESPI is 0 or [GICD_TYPER](#).NMI is 0, these registers are RES0.

When [GICD_TYPER](#).ESPI is 1: the number of implemented GICD_INMIR<n>E registers is ([GICD_TYPER](#).ESPI range+1). Registers are numbered from 0.

GICD_INMIR<n>E is a 32-bit register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
NMI31	NMI30	NMI29	NMI28	NMI27	NMI26	NMI25	NMI24	NMI23	NMI22	NMI21	NMI20	NMI19	NMI18	NMI17	NMI16	NMI15	NMI14

NMI<x>	Meaning
0b0	Interrupt does not have the non-maskable property.
0b1	Interrupt has the non-maskable property.

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than one time. The effect of the change must be visible in finite time.

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x3B00 + (4 * n)	GICD_INMIR<n>E

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

GICD_IROUTER<n>, Interrupt Routing Registers, n = 32 - 1019

The GICD_IROUTER<n> characteristics are:

Purpose

When affinity routing is enabled, provides routing information for the SPI with INTID n.

Configuration

These registers are available in all configurations of the GIC. If the GIC implementation supports two Security states, these registers are Common.

The maximum value of n is given by (32*(GICD_TYPER.ITLinesNumber+1) - 1). [GICD_IROUTER<n>](#) registers where n=0 to 31 are reserved.

Attributes

GICD_IROUTER<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0																								Aff3								
Interrupt_Routing_Mode	RES0								Aff2								Aff1								Aff0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

Affinity level 3.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Interrupt_Routing_Mode, bit [31]

Interrupt Routing Mode. Defines how SPIs are routed in an affinity hierarchy:

Interrupt_Routing_Mode	Meaning
0b0	Interrupts routed to the PE specified by a.b.c.d. In this routing, a, b, c, and d are the values of fields Aff3, Aff2, Aff1, and Aff0 respectively.
0b1	Interrupts routed to any PE defined as a participating node.

If GICD_IROUTER<n>.IRM == 0 and the affinity path does not correspond to an implemented PE, then if the corresponding interrupt becomes pending behavior is CONSTRAINED UNPREDICTABLE:

- The interrupt is not forwarded to any PE, direct reads return the written value
- The affinity path is treated as an UNKNOWN implemented PE, direct reads return the UNKNOWN implemented PE
- The affinity path is treated as an UNKNOWN implemented PE, direct reads return the written value

WhenIn implementations that do not require 1 of N distribution of SPIs, this bit might be RAZ/WI. GICD_TYPER.No1N is 1, 1 of N distribution is not supported. Setting this field to 1 is CONSTRAINED UNPREDICTABLE, the permitted behaviors are:

- The field behaves as if set to 0 for all purposes.
- The field behaves as if set to 0 for all purposes other than a direct-read of the register.
- The interrupt is treated as not targeting any PE.

When this bit is set to 1, GICD_IROUTER<n>.{Aff3, Aff2, Aff1, Aff0} are UNKNOWN.

Note

An implementation might choose to make the Aff<n> fields RO when this field is 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Bits [30:24]

Reserved, RES0.

Aff2, bits [23:16]

Affinity level 2.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Aff1, bits [15:8]

Affinity level 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Aff0, bits [7:0]

Affinity level 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

For an SPI with INTID m:

- The corresponding GICD_IROUTER<n> register number, n, is given by $n = m$.
- The offset of the GICD_IROUTER<n> register is $0 \times 6000 + 8n$.

Accessing GICD_IROUTER<n>

These registers are used only when affinity routing is enabled. When affinity routing is not enabled:

- These registers are RES0. An implementation is permitted to make the register RAZ/WI in this case.
- The `GICD_ITARGETSR<n>` registers provide interrupt routing information.

Note

When affinity routing becomes enabled for a Security state (for example, following a reset or following a write to [GICD_CTLR](#)) the value of all writeable fields in this register is UNKNOWN for that Security state. When the group of an interrupt changes so the ARE setting for the interrupt changes to 1, the value of this register is UNKNOWN for that interrupt.

If `GICD_CTLR.DS==0`, unless the `GICD_NSACR<n>` registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any `GICD_IROUTER<n>` registers that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

Note

For each interrupt, a GIC implementation might support fewer than 256 values for an affinity level. In this case, some bits of the corresponding affinity level field might be RO. Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than one time. The effect of the change must be visible in finite time.

GICD_IROUTER<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x6000 + (8 * n)	GICD_IROUTER<n>

Accesses to this interface are **RW**.

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The GICD_IROUTER<n>E characteristics are:

Purpose

When affinity routing is enabled, provides routing information for the corresponding SPI in the extended SPI range.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_IROUTER<n>E are RES0.

When [GICD_TYPER](#).ESPI==0, these registers are RES0.

When [GICD_TYPER](#).ESPI==1, the number of implemented GICD_IROUTER<n>E registers is ((([GICD_TYPER](#).ESPI range+1)*32)-1). Registers are numbered from 0.

Attributes

GICD_IROUTER<n>E is a 64-bit register.

Field descriptions

63										62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32																																							
										RES0																						Aff3																	
Interrupt_Routing_Mode										RES0										Aff2										Aff1										Aff0									
31										30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																																							

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

Affinity level 3.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Interrupt Routing Mode, bit [31]

Interrupt Routing Mode. Defines how SPIs are routed in an affinity hierarchy:

Interrupt	Routing Mode	Meaning
	0b0	Interrupts routed to the PE specified by a.b.c.d. In this routing, a, b, c, and d are the values of fields Aff3, Aff2, Aff1, and Aff0 respectively.
	0b1	Interrupts routed to any PE defined as a participating node.

If GICD_IROUTER<n>E.IRM == 0 and the affinity path does not correspond to an implemented PE, then if the corresponding interrupt becomes pending behavior is CONSTRAINED UNPREDICTABLE:

- The interrupt is not forwarded to any PE, direct reads return the written value
- The affinity path is treated as an UNKNOWN implemented PE, direct reads return the UNKNOWN implemented PE
- The affinity path is treated as an UNKNOWN implemented PE, direct reads return the written value

When ~~In implementations that do not require 1 of N distribution of SPIs, this bit might be RAZ/WI.~~ GICD_TYPER.No1N is 1, 1 of N distribution is not supported. Setting this field to 1 is CONSTRAINED UNPREDICTABLE, the permitted behaviors are:

- The field behaves as if set to 0 for all purposes.
- The field behaves as if set to 0 for all purposes other than a direct-read of the register.
- The interrupt is treated as not targeting any PE.

When this bit is set to 1, GICD_IROUTER<n>E.{Aff3, Aff2, Aff1, Aff0} are UNKNOWN.

Note

An implementation might choose to make the Aff<n> fields RO when this field is 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Bits [30:24]

Reserved, RES0.

Aff2, bits [23:16]

Affinity level 2.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Aff1, bits [15:8]

Affinity level 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

Aff0, bits [7:0]

Affinity level 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to an architecturally UNKNOWN value.

For an SPI with INTID m:

- The corresponding GICD_IROUTER<n>E register number, n, is given by $n = m$.
- The offset of the GICD_IROUTER<n>E register is $0 \times 6000 + 8n$.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_IROUTER<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x8000 + (8 * n)	GICD_IROUTER<n>E

Accesses to this interface are **RW**.

(old)

htmldiff from-

(new)

GICR_INMIR0, Non-maskable Interrupt Register for PPIs.

The GICR_INMIR0 characteristics are:

Purpose

Controls whether the corresponding PPI has the non-maskable property.

Configuration

This register is present only when FEAT_GICv3_NMI is implemented. Otherwise, direct accesses to GICR_INMIR0 are RES0.

When [GICD_TYPER.NMI](#) is 0, this register is RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_INMIR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
nmi31	nmi30	nmi29	nmi28	nmi27	nmi26	nmi25	nmi24	nmi23	nmi22	nmi21	nmi20	nmi19	nmi18	nmi17	nmi16	nmi15	nmi14

nmi<x>, bit [x], for x = 31 to 0

Non-maskable property.

nmi<x>	Meaning
0b0	Interrupt does not have the non-maskable property.
0b1	Interrupt has the non-maskable property.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

If affinity routing is disabled for the Security state of an interrupt, the bit is RES0.

This bit is RES0 when the corresponding interrupt is configured as Group 0.

Accessing GICR_INMIR0

Bits corresponding to unimplemented interrupts are RAZ/WI.

When [GICD_CTLR.DS](#)=0, bits corresponding to Group 0 and Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0F80	GICR_INMIR0

3005/0907/2022 1517:5807; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

The GICR_INMIR<n>E characteristics are:

Controls whether the corresponding Extended PPI has the non-maskable property.

This register is present only when FEAT_GICv3p1 is implemented and FEAT_GICv3_NMI is implemented. Otherwise, direct accesses to GICR_INMIR<n>E are RES0.

When [GICR_TYPER.PPINum](#) is 0b0000 or [GICD_TYPER.NMI](#) is 0, these registers are RES0.

A copy of this register is provided for each Redistributor.

GICR_INMIR<n>E is a 32-bit register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
nmi31	nmi30	nmi29	nmi28	nmi27	nmi26	nmi25	nmi24	nmi23	nmi22	nmi21	nmi20	nmi19	nmi18	nmi17	nmi16	nmi15	nmi14

nmi<x>, bit [x], for x = 31 to 0

Non-maskable property.

nmi<x>	Meaning
0b0	Interrupt does not have the non-maskable property.
0b1	Interrupt has the non-maskable property.

This bit is RES0 when the corresponding interrupt is configured as Group 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

If affinity routing is disabled for the Security state of an interrupt, the bit is RES0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

When [GICD_CTLR.DS==0](#), bits corresponding to Group 0 and Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0F80 + (4 * n)	GICR_INMIR<n>E

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

(old)	htmldiff from-	(new)
-------	----------------	-------

VSGI, bit [26]**When FEAT_GICv4p1 is implemented:**

Indicates whether vSGIs are supported.

VSGI	Meaning
0b0	Direct injection of SGIs not supported.
0b1	Direct injection of SGIs supported.

Otherwise:

Reserved, RES0.

CommonLPIAff, bits [25:24]

Indicates the affinity scope level of at the which CommonLPIAff Redistributors group share an LPI Configuration table.

CommonLPIAff	Meaning
0b00	All Redistributors are must members share of an the LPI same Configuration CommonLPIAff group table.
0b01	All Redistributors with the same Aff3 value are must members share of an the LPI same Configuration CommonLPIAff group table.
0b10	All Redistributors with the same Aff3.Aff2 value are must members share of an the LPI same Configuration CommonLPIAff group table.
0b11	All Redistributors with the same Aff3.Aff2.Aff1 value are must members share of an the LPI same Configuration CommonLPIAff group table.

Redistributors in the same CommonLPIAff group must use the same copy of the LPI Configuration table, and if GICv4.1 is implemented the same copy of the vPE Configuration table.

Processor_Number, bits [23:8]

A unique identifier for the PE. When [GITS_TYPER.PTA](#) == 0, an ITS uses this field to identify the interrupt target.

When affinity routing is disabled for a Security state, this field indicates which [GICD_ITARGETSR<n>](#) corresponds to this Redistributor.

RVPEID, bit [7]**When FEAT_GICv4p1 is implemented:**

Indicates how the resident vPE is specified.

RVPEID	Meaning
0b0	GICR_VPENDBASER records the address of the vPE's Virtual Pending Table.
0b1	GICR_VPENDBASER records vPEID.

Otherwise:

Reserved, RES0.

MPAM, bit [6]**When FEAT_GICv3p1 is implemented:**

MPAM

MPAM	Meaning
0b0	MPAM not supported.
0b1	MPAM supported.

Otherwise:

Reserved, RES0.

DPGS, bit [5]

Sets support for [GICR_CTLR.DPG*](#) bits.

DPGS	Meaning
0b0	GICR_CTLR.DPG* bits are not supported.
0b1	GICR_CTLR.DPG* bits are supported.

Last, bit [4]

Indicates whether this Redistributor is the highest-numbered Redistributor in a series of contiguous Redistributor pages.

Last	Meaning
0b0	This Redistributor is not the highest-numbered Redistributor in a series of contiguous Redistributor pages.
0b1	This Redistributor is the highest-numbered Redistributor in a series of contiguous Redistributor pages.

DirectLPI, bit [3]

Indicates whether this Redistributor supports direct injection of LPIs.

DirectLPI	Meaning
0b0	This Redistributor does not support direct injection of LPIs. The GICR_SETLPIR , GICR_CLRLPIR , GICR_INVLPIR , GICR_INVALLR , and GICR_SYNCRR registers are either not implemented, or have an IMPLEMENTATION DEFINED purpose.
0b1	This Redistributor supports direct injection of LPIs. The GICR_SETLPIR , GICR_CLRLPIR , GICR_INVLPIR , GICR_INVALLR , and GICR_SYNCRR registers are implemented.

Dirty, bit [2]

Controls the functionality of [GICR_VPENDBASER.Dirty](#).

Dirty	Meaning
0b0	GICR_VPENDBASER.Dirty is UNKNOWN when GICR_VPENDBASER.Valid == 1.
0b1	GICR_VPENDBASER.Dirty indicates when the Virtual Pending Table has been parsed when GICR_VPENDBASER.Valid is written from 0 to 1.

When [GICR_TYPER.VLPIS](#) == 0, this field is RES0.

Note

In GICv4p1 implementations this field is RES1.

(old)htmldiff from-(new)

MPAMCFG_DIS, MPAM Partition Configuration Disable Register

The MPAMCFG_DIS characteristics are:

Purpose

Disables a PARTID configuration as set in other MPAMCFG registers.

MPAMCFG_DIS_s disables a Secure PARTID. MPAMCFG_DIS_ns disables a Non-secure PARTID. MPAMCFG_DIS_rl disables a Realm PARTID. MPAMCFG_DIS_rt disables a Root PARTID.

Configuration

The power domain of MPAMCFG_DIS is IMPLEMENTATION DEFINED.

This register is present only when (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_ENDIS == 1. Otherwise, direct accesses to MPAMCFG_DIS are RES0.

Attributes

MPAMCFG_DIS is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NFU																PARTID															

NFU, bit [31]
When MPAMF_IDR.HAS_NFU == 1:

No Future Use. Software indicates that the PARTID disabled with NFU of 1 will not be used again and will be reconfigured and reenabled before being used again.

NFU	Meaning
0b0	Control settings of the disabled PARTID must be retained.
0b1	Control settings of the disabled PARTID may take an UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [30:16]

Reserved, RES0.

PARTID, bits [15:0]

Selects the PARTID to disable.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

(old)

htmldiff from-

(new)

MPAMCFG_PART_SEL, MPAM Partition Configuration Selection Register

The MPAMCFG_PART_SEL characteristics are:

Purpose

Selects a partition ID to configure.

MPAMCFG_PART_SEL_s selects a Secure PARTID to configure. MPAMCFG_PART_SEL_ns selects a Non-secure PARTID to configure. MPAMCFG_PART_SEL_rt selects a Root PARTID to configure. MPAMCFG_PART_SEL_rl selects a Realm PARTID to configure.

After setting this register with a PARTID, software (usually a hypervisor) can perform a series of accesses to MPAMCFG registers to configure parameters for MPAM resource controls to use when requests have that PARTID.

Configuration

The power domain of MPAMCFG_PART_SEL is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and (MPAMF_IDR.HAS_CCAP_PART == 1, or MPAMF_IDR.HAS_CPOR_PART == 1, or MPAMF_IDR.HAS_MBW_PART == 1, or MPAMF_IDR.HAS_PRI_PART == 1, or MPAMF_IDR.HAS_PARTID_NRW == 1, or (MPAMF_IDR.EXT == 0 and MPAMF_IDR.HAS_IMPL_IDR == 1) or (MPAMF_IDR.EXT == 1, MPAMF_IDR.HAS_IMPL_IDR == 1 and MPAMF_IDR.NO_IMPL_PART == 0)). Otherwise, direct accesses to MPAMCFG_PART_SEL are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_PART_SEL is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				RIS				RES0				INTERNAL				PARTID_SEL															

Bits [31:28]

Reserved, RES0.

RIS, bits [27:24]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented), MPAMF_IDR.EXT == 1 and MPAMF_IDR.HAS_RIS == 1:

Resource Instance Selector. RIS selects one resource to configure through MPAMCFG registers and describe with MPAMF ID registers.

Otherwise:

Reserved, RES0.

Bits [23:17]

Reserved, RES0.

INTERNAL, bit [16]

Internal PARTID.

If [MPAMF_IDR.HAS_PARTID_NRW = 1](#):

If [MPAMF_IDR.HAS_PARTID_NRW == 0](#), this field is RAZ/WI. **0b1**:

INTERNAL	Meaning
0b0	PARTID_SEL is interpreted as a request PARTID and ignored except for use with MPAMCFG_INTPARTID register access.
0b1	PARTID_SEL is interpreted as an internal PARTID and used for access to MPAMCFG control settings except for MPAMCFG_INTPARTID .

If PARTID narrowing is implemented as indicated by [MPAMF_IDR.HAS_PARTID_NRW = 1](#), when accessing other MPAMCFG registers the value of the MPAMCFG_PART_SEL.INTERNAL bit is checked for these conditions:

- When the [MPAMCFG_INTPARTID](#) register is read or written, if the value of MPAMCFG_PART_SEL.INTERNAL is not 0, an Unexpected_INTERNAL error is set in [MPAMF_ESR](#).
- When an MPAMCFG register other than [MPAMCFG_INTPARTID](#) is read or written, if the value of MPAMCFG_PART_SEL.INTERNAL is not 1, [MPAMF_ESR](#) is set to indicate an intPARTID_Range error.

In either error case listed here, the value returned by a read operation is UNPREDICTABLE, and the control settings are not affected by a write.

When [MPAMF_IDR.HAS_PARTID_NRW == 0](#), access to this field is **RAZ/WI**.

PARTID_SEL, bits [15:0]

Selects the partition ID to configure.

Reads and writes to other MPAMCFG registers are indexed by PARTID_SEL and by the NS bit used to access MPAMCFG_PART_SEL to access the configuration for a single partition.

Accessing MPAMCFG_PART_SEL

This register is within the MPAM feature page memory frames. In a system that supports Secure and Non-secure memory maps, there must be both Secure and Non-secure MPAM feature pages.

MPAMCFG_PART_SEL_s must only be accessible from the Secure MPAM feature page. MPAMCFG_PART_SEL_ns must only be accessible from the Non-secure MPAM feature page.

MPAMCFG_PART_SEL_s and MPAMCFG_PART_SEL_ns must be separate registers. The Secure instance (MPAMCFG_PART_SEL_s) accesses the PARTID selector used for Secure PARTIDs, and the Non-secure instance (MPAMCFG_PART_SEL_ns) accesses the PARTID selector used for Non-secure PARTIDs.

MPAMCFG_PART_SEL can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0100	MPAMCFG_PART_SEL_s

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0100	MPAMCFG_PART_SEL_ns

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0100	MPAMCFG_PART_SEL_rt

When FEAT_RME is implemented, accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0100	MPAMCFG_PART_SEL_rl

When FEAT_RME is implemented, accesses to this interface are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_ERR_MSI_ADDR_H, MPAM Error MSI High-part Address Register

The MPAMF_ERR_MSI_ADDR_H characteristics are:

Purpose

MPAMF_ERR_MSI_ADDR_H is a 32-bit read/write register for the high part of the MPAM error MSI address.

MPAMF_ERR_MSI_ADDR_H_s is the high part of the MSI write address for error interrupts related to Secure PARTIDs. MPAMF_ERR_MSI_ADDR_H_ns is the high part of the MSI write address for error interrupts related to Non-secure PARTIDs. MPAMF_ERR_MSI_ADDR_H_rt is the high part of the MSI write address for error interrupts related to Root PARTIDs. MPAMF_ERR_MSI_ADDR_H_rl is the high part of the MSI write address for error interrupts related to Realm PARTIDs.

Configuration

The power domain of MPAMF_ERR_MSI_ADDR_H is IMPLEMENTATION DEFINED.

This register is present only when (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_ERR_MSI == 1. Otherwise, direct accesses to MPAMF_ERR_MSI_ADDR_H are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ERR_MSI_ADDR_H is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												MSI_ADDR_H																			

Bits [31:20]

Reserved, RES0.

MSI_ADDR_H, bits [19:0]

MSI write address bits[51:32].

Accessing MPAMF_ERR_MSI_ADDR_H

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MPAMF_ERR_MSI_ADDR_H_s must only be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_ADDR_H_ns must only be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_ADDR_H_rt must only be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_ADDR_H_rl must only be accessible from the Realm MPAM feature page.

MPAMF_ERR_MSI_ADDR_H_s, MPAMF_ERR_MSI_ADDR_H_ns, MPAMF_ERR_MSI_ADDR_H_rt, and MPAMF_ERR_MSI_ADDR_H_rl must be separate registers:

- The Secure instance (MPAMF_ERR_MSI_ADDR_H_s) accesses the high part of the memory address for MSI write to signal an MPAM error used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ERR_MSI_ADDR_H_ns) accesses the high part of the memory address for MSI write to signal an MPAM error used for Non-secure PARTIDs.
- The Root instance (MPAMF_ERR_MSI_ADDR_H_rt) accesses the high part of the memory address for MSI write to signal an MPAM error used for Root PARTIDs.
- The Realm instance (MPAMF_ERR_MSI_ADDR_H_rl) accesses the high part of the memory address for MSI write to signal an MPAM error used for Realm PARTIDs.

MPAMF_ERR_MSI_ADDR_H can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00E4	MPAMF_ERR_MSI_ADDR_H_s

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00E4	MPAMF_ERR_MSI_ADDR_H_ns

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00E4	MPAMF_ERR_MSI_ADDR_H_rt

When FEAT_RME is implemented, accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00E4	MPAMF_ERR_MSI_ADDR_H_rl

When FEAT_RME is implemented, accesses to this interface are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

- The Secure instance (MPAMF_ERR_MSI_ADDR_L_s) accesses the low part of the memory address for MSI write to signal an MPAM error used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ERR_MSI_ADDR_L_ns) accesses the low part of the memory address for MSI write to signal an MPAM error used for Non-secure PARTIDs.
- The Root instance (MPAMF_ERR_MSI_ADDR_L_rt) accesses the low part of the memory address for MSI write to signal an MPAM error used for Root PARTIDs.
- The Realm instance (MPAMF_ERR_MSI_ADDR_L_rl) accesses the low part of the memory address for MSI write to signal an MPAM error used for Realm PARTIDs.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00E0	MPAMF_ERR_MSI_ADDR_L_s

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00E0	MPAMF_ERR_MSI_ADDR_L_ns

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00E0	MPAMF_ERR_MSI_ADDR_L_rt

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x00E0	MPAMF_ERR_MSI_ADDR_L_r1

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Note: This encoding matches the VMSAv8-64 stage 2 MemAttr[3:0] field as described in the Arm ARM, except that the following encodings are Reserved (not UNPREDICTABLE) and behave as Device-nGnRnE: 0b0100, 0b1000, and 0b1100.

MSI_MEMATTR	Meaning
0b0000	Device-nGnRnE.
0b0001	Device-nGnRE.
0b0010	Device-nGRE.
0b0011	Device-GRE.
0b0100	Reserved. Behave as Device-nGnRnE, 0b0000.
0b0101	Normal Inner Non-cacheable, Outer Non-cacheable.
0b0110	Normal Inner Write-Through Cacheable, Outer Non-cacheable.
0b0111	Normal Inner Write-Back Cacheable, Outer Non-cacheable.
0b1000	Reserved. Behave as Device-nGnRnE, 0b0000.
0b1001	Normal Inner Non-Cachable, Outer Write-Through Cacheable.
0b1010	Normal Inner Write-Through Cacheable, Outer Write-Through Cacheable.
0b1011	Normal Inner Write-Back Cacheable, Outer Write-Through Cacheable.
0b1100	Reserved. Behave as Device-nGnRnE, 0b0000.
0b1101	Normal Inner Non-cacheable, Outer Write-Back Cacheable.
0b1110	Normal Inner Write-Through Cacheable, Outer Write-Back Cacheable.
0b1111	Normal Inner Write-Back Cacheable, Outer Write-Back Cacheable.

When this field specifies a Device memory type, the contents of MPAMF_ERR_MSI_ATTR.MSI_SH are IGNORED and Shareability is effectively Outer Shareable.

Device types may be implemented as any Device type with more than 'n' characters. For example, if this field is set to 0b0010, an implementation may treat the MSI write as the specified type, Device-nGRE, or as Device-nGnRE or as Device-nGnRnE.

Reserved encodings 0b0100, 0b1000, and 0b1100 must be implemented to behave the same as the 0b0000 encoding.

Bits [23:1]

Reserved, RES0.

MSIEN, bit [0]

Error interrupt MSI Enable.

MSIEN	Meaning
0b0	MPAM error MSI writes are not generated to signal enabled MPAM error interrupts. When error MSI writes are disabled, hardwired error interrupts could be generated.
0b1	MPAM error MSI writes are generated to signal enabled MPAM error interrupts. When error MSI writes are enabled, hardwired error interrupts are not generated.

The value of this field affects whether hardwired error interrupts are generated.

The reset behavior of this field is:

- On a MSC reset, this field resets to 0.

Accessing MPAMF_ERR_MSI_ATTR

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MPAMF_ERR_MSI_ATTR_s must only be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_ATTR_ns must only be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_ATTR_rt must only be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_ATTR_rl must only be accessible from the Realm MPAM feature page.

MPAMF_ERR_MSI_ATTR_s, MPAMF_ERR_MSI_ATTR_ns, MPAMF_ERR_MSI_ATTR_rt, and MPAMF_ERR_MSI_ATTR_rl must be separate registers:

- The Secure instance (MPAMF_ERR_MSI_ATTR_s) accesses the memory access attributes for MSI write to signal an MPAM error used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ERR_MSI_ATTR_ns) accesses the memory access attributes for MSI write to signal an MPAM error used for Non-secure PARTIDs.
- The Root instance (MPAMF_ERR_MSI_ATTR_rt) accesses the memory access attributes for MSI write to signal an MPAM error used for Root PARTIDs.
- The Realm instance (MPAMF_ERR_MSI_ATTR_rl) accesses the memory access attributes for MSI write to signal an MPAM error used for Realm PARTIDs.

MPAMF_ERR_MSI_ATTR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF BASE s	0x00EC	MPAMF ERR MSI ATTR s

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE ns	0x00EC	MPAMF_ERR MSI ATTR ns

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00EC	MPAMF_ERR_MSI_ATTR_rt

When FEAT RME is implemented, accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x00EC	MPAMF_ERR_MSI_ATTR_r1

When FEAT RME is implemented, accesses to this interface are **RW**.

3005/0907/2022 15:17:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_ERR_MSI_DATA, MPAM Error MSI Data Register

The MPAMF_ERR_MSI_DATA characteristics are:

Purpose

MPAMF_ERR_MSI_DATA is a 32-bit read/write register for the MPAM error MSI data.

MPAMF_ERR_MSI_DATA_s is the data for the MSI write for error interrupts related to Secure PARTIDs.
 MPAMF_ERR_MSI_DATA_ns is the data for the MSI write for error interrupts related to Non-secure PARTIDs.
 MPAMF_ERR_MSI_DATA_rt is the data for the MSI write for error interrupts related to Root PARTIDs.
 MPAMF_ERR_MSI_DATA_rl is the data for the MSI write for error interrupts related to Realm PARTIDs.

Configuration

The power domain of MPAMF_ERR_MSI_DATA is IMPLEMENTATION DEFINED.

This register is present only when (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_ERR_MSI == 1. Otherwise, direct accesses to MPAMF_ERR_MSI_DATA are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ERR_MSI_DATA is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																MSI_DATA															

MSI_DATA, bits [31:0]

MSI data to be written to ITS to signal an MSI.

Accessing MPAMF_ERR_MSI_DATA

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MPAMF_ERR_MSI_DATA_s must only be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_DATA_ns must only be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_DATA_rt must only be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_DATA_rl must only be accessible from the Realm MPAM feature page.

MPAMF_ERR_MSI_DATA_s, MPAMF_ERR_MSI_DATA_ns, MPAMF_ERR_MSI_DATA_rt, and MPAMF_ERR_MSI_DATA_rl must be separate registers:

- The Secure instance (MPAMF_ERR_MSI_DATA_s) accesses the data for MSI write to signal an MPAM error used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ERR_MSI_DATA_ns) accesses the data for MSI write to signal an MPAM error used for Non-secure PARTIDs.
- The Root instance (MPAMF_ERR_MSI_DATA_rt) accesses the data for MSI write to signal an MPAM error used for Root PARTIDs.

- The Realm instance (MPAMF_ERR_MSI_DATA_rl) accesses the data for MSI write to signal an MPAM error used for Realm PARTIDs.

MPAMF_ERR_MSI_DATA can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00E8	MPAMF_ERR_MSI_DATA_s

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00E8	MPAMF_ERR_MSI_DATA_ns

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00E8	MPAMF_ERR_MSI_DATA_rt

When FEAT_RME is implemented, accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x00E8	MPAMF_ERR_MSI_DATA_r1

When FEAT_RME is implemented, accesses to this interface are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MPAMF_ERR_MSI_MPAM_s, MPAMF_ERR_MSI_MPAM_ns, MPAMF_ERR_MSI_MPAM_rt, and MPAMF_ERR_MSI_MPAM_rl must be separate registers:

- | Component | Frame | Offset | Instance |
|-----------|--------------|--------|----------------------|
| MPAM | MPAMF_BASE_s | 0x00DC | MPAMF_ERR_MSI_MPAM_s |

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00DC	MPAMF_ERR_MSI_MPAM_ns

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00DC	MPAMF_ERR_MSI_MPAM_rt

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00DC	MPAMF_ERR_MSI_MPAM_rl

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

MSMON_CAPT_EVNT, MPAM Capture Event Generation Register

The MSMON_CAPT_EVNT characteristics are:

Purpose

Generates a local capture event when written with bit[0] as 1.

MSMON_CAPT_EVNT_s generates local capture events for Secure monitor instances only or for Secure and Non-secure monitor instances. MSMON_CAPT_EVNT_ns generates local capture events for Non-secure monitor instances only. MSMON_CAPT_EVNT_rt generates local capture events for Root monitor instances only or for Root, Secure, Realm, and Non-secure monitor instances. MSMON_CAPT_EVNT_rl generates local capture events for Realm monitor instances or for for Realm monitor instances or Realm and Non-secure monitor instances.

Configuration

The power domain of MSMON_CAPT_EVNT is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1 and MPAMF_MSMON_IDR.HAS_LOCAL_CAPT_EVNT == 1. Otherwise, direct accesses to MSMON_CAPT_EVNT are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CAPT_EVNT is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																ALL		NOW													

Bits [31:2]

Reserved, RES0.

ALL, bit [1]

In the Secure instance of this register:

- If ALL is written as 1 and NOW is also written as 1, signal a capture event to Secure and Non-secure monitor instances in this MSC that are configured with CAPT_EVNT = 7.
- If ALL is written as 0 and NOW is written as 1, signal a capture event to Secure monitor instances in this MSC that are configured with CAPT_EVNT = 7.

In the Non-secure instance of this register, this **field** is RAZ/WI.

In the Root instance of this register:

- If ALL is written as 1 and NOW is also written as 1, signal a capture event to Root, Realm, Secure, and Non-secure monitor instances in this MSC that are configured with CAPT_EVNT = 7.
- If ALL is written as 0 and NOW is written as 1, signal a capture event to Root monitor instances within this MSC that are configured with CAPT_EVNT = 7.

In the Realm instance of this register:

- If ALL is written as 1 and NOW is also written as 1, signal a capture event to Realm and Non-secure monitor instances in this MSC that are configured with CAPT_EVNT = 7.
- If ALL is written as 0 and NOW is written as 1, signal a capture event to Realm monitor instances within this MSC that are configured with CAPT_EVNT = 7.

This bit always reads as zero.

ALL	Meaning
0b0	Send capture event only to monitor instances in the same MPAM feature page as this register.
0b1	Send capture event to monitor instances in certain MPAM feature pages as described in the ALL field of this register.

NOW, bit [0]

When written as 1, this bit causes an event to those monitor instances described in the ALL field that have CAPT_EVNT set to the value of 7.

When this bit is written as 0, no event is signaled.

This bit always reads as zero.

Accessing MSMON_CAPT_EVNT

This register is within the MPAM feature page memory frames. In a system that supports Secure and Non-secure memory maps, there must be both Secure and Non-secure MPAM feature pages.

MSMON_CAPT_EVNT_s must only be accessible from the Secure MPAM feature page. MSMON_CAPT_EVNT_ns must only be accessible from the Non-secure MPAM feature page.

MSMON_CAPT_EVNT_s and MSMON_CAPT_EVNT_ns must be separate registers. The Secure instance (MSMON_CAPT_EVNT_s) can generate local capture events for Secure monitor instances only or for Secure and Non-secure monitor instances, and the Non-secure instance (MSMON_CAPT_EVNT_ns) can generate local capture events for Non-secure monitor instances only.

MSMON_CAPT_EVNT can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0808	MSMON_CAPT_EVNT_s

Accesses to this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0808	MSMON_CAPT_EVNT_ns

Accesses to this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0808	MSMON_CAPT_EVNT_rt

When FEAT_RME is implemented, accesses to this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0808	MSMON_CAPT_EVNT_rl

When FEAT_RME is implemented, accesses to this interface are **WO/RAZ**.

(old)**htmldiff from-****(new)**

The MSMON_CFG_CSU_CTL characteristics are:

Purpose

Controls the CSU monitor selected by [MSMON_CFG_MON_SEL](#).

MSMON_CFG_CSU_CTL_s controls the Secure cache storage usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_CTL_ns controls Non-secure cache storage usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_CTL_rt controls the monitor configuration for the Root PARTID selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_CTL_rl controls the monitor configuration for the Realm PARTID selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If `MPAMF_IDR.HAS_RIS` is 1, the monitor instance configuration accessed is for the resource instance currently selected by `MSMON_CFG_MON_SEL.RIS` and the monitor instance of that resource instance selected by `MSMON_CFG_MON_SEL.MON_SEL`.

Configuration

The power domain of MSMON_CFG_CSU_CTL is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1 and MPAMF_MSMON_IDR.MSMON_CSU == 1. Otherwise, direct accesses to MSMON_CFG_CSU_CTL are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON CFG CSU CTL is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18
ENCAPT_EVNT				CAPT_RESET		OFLOW_STATUS		OFLOW_INTR		OFLOW_FRZ		OFLOW_CAPT	
SUBTYPE				RES0		CEVNT		OFLW		MAT			

EN, bit [31]

Enabled.

EN	Meaning
0b0	The monitor instance is disabled and must not collect any information.
0b1	The monitor instance is enabled to collect information according to the configuration of the instance.

CAPT_EVNT, bits [30:28]

Capture event selector.

Select the event that triggers capture from the following:

CAPT_EVNT	Meaning
0b000	No capture event is triggered.
0b001	External capture event 1 (optional, but recommended)
0b010	External capture event 2 (optional)
0b011	External capture event 3 (optional)
0b100	External capture event 4 (optional)
0b101	External capture event 5 (optional)
0b110	External capture event 6 (optional)
0b111	Capture occurs when a MSMON_CAPT_EVNT register in this MSC is written and causes a capture event for the Security state of this monitor. (optional)

The values marked as optional indicate capture event sources that can be omitted in an implementation. Those values representing non-implemented event sources must not trigger a capture event.

When MPAMF_CSUMON_IDR.HAS_CAPTURE == 0, access to this field is **RAZ/WI**.

CAPT_RESET, bit [27]

Reset after capture.

Controls whether the value of [MSMON_CSU](#) is reset to zero immediately after being copied to [MSMON_CSU_CAPTURE](#).

CAPT_RESET	Meaning
0b0	Monitor is not reset on capture.
0b1	Monitor is reset on capture.

Because the CSU monitor type produces a measurement rather than a count, it might not make sense to ever reset the value after a capture. If there is no reason to ever reset a CSU monitor, this field is RAZ/WI.

When MPAMF_CSUMON_IDR.HAS_CAPTURE == 0, access to this field is **RAZ/WI**.

OFLOW_STATUS, bit [26]

Overflow status.

Indicates whether the value of [MSMON_CSU](#) has overflowed.

If [MPAMF_CSUMON_IDR.HAS_CEVNT_OFLW](#) is 1 or [MPAMF_CSUMON_IDR.HAS_OFLOW_LNKG](#) is 1, then a store to [MSMON_CSU](#) when this field is 1 resets this field to 0.

OFLOW_STATUS	Meaning
0b0	No overflow has occurred.
0b1	At least one overflow has occurred since this bit was last written to zero.

If overflow is not possible for a CSU monitor in the implementation, this field is RAZ/WI.

OFLOW_INTR, bit [25]

Overflow Interrupt.

Controls whether an overflow interrupt is generated when the value of [MSMON_CSU](#) has overflowed.

OFLOW_INTR	Meaning
0b0	No interrupt is signaled on an overflow of MSMON_CSU .
0b1	On overflow, an implementation-specific interrupt is signaled.

WhenIf MSMON_CFG_CSU_CTL.OFLOW_INTR is 0, not supported by the implementation, this field is **RAZ/WI**.

OFLOW_FRZ, bit [24]

Freeze Monitor on Overflow.

Controls whether the value of [MSMON_CSU.VALUE](#) freezes on an overflow.

OFLOW_FRZ	Meaning
0b0	Monitor count wraps on overflow.
0b1	Monitor count freezes on overflow. The frozen value might be 0 or another value if the monitor overflowed with an increment larger than 1.

If overflow is not possible for a CSU monitor in the implementation, this field is RAZ/WI.

When a [MSMON_CSU.VALUE](#) of a monitor instance is frozen it does not change until [MSMON_CSU](#) register for that instance has been written.

OFLOW_CAPT, bit [23]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_OFLOW_CAPT == 1:

Capture Monitor on Overflow.

OFLOW_CAPT	Meaning
0b0	Monitor is not captured on an overflow or when affected by an overflow linkage event.
0b1	Monitor is captured and the MSMON_CSU .{NRDY, VALUE} fields are copied to the monitor instance's capture register on an overflow or when affected by an overflow linkage event. The monitor instance treats an overflow of this monitor instance as a private capture event. If MSMON_CFG_MBWU_CTL.CEVNT_OFLW is 1, this monitor instance also treats an overflow linkage event as a capture event. If the OFLOW_FRZ field is 1, the monitor does not continue to count after the overflow or overflow linkage event. If the CAPT_RESET field is 1, the monitor instance resets to 0.

Otherwise:

Reserved, RES0.

SUBTYPE, bits [22:20]

Subtype. Type of cache storage usage counted by this monitor.

This field is not currently used for CSU monitors, but reserved for future use.

This field is RAZ/WI.

Bit [19]

Reserved, RES0.

CEVNT_OFLW, bit [18]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_CEVNT_OFLW == 1:

Capture Event performs overflow behavior.

CEVNT_OFLW	Meaning
0b0	On a capture event matching the CAPT_EVNT field, the capture behaviors are performed. The MSMON_CSU .{VALUE, NRDY} fields are transferred to the monitor instance's capture register.
0b1	On a capture event matching the CAPT_EVNT field, the monitor instance treats a capture event as an overflow and the overflow behaviors are performed. The behavior is controlled by the MSMON_CFG_CSU_CTL .{OFLOW_FRZ, OFLOW_CAPT, CAPT_RESET} fields. The MSMON_CFG_CSU_CTL .OFLOW_STATUS field is set for this monitor instance.

Otherwise:

Reserved, RES0.

MATCH_PMG, bit [17]

Match PMG.

Controls whether the monitor measures only storage used with PMG matching [MSMON_CFG_CSU_FLT](#).PMG.

MATCH_PMG	Meaning
0b0	The monitor measures storage used with any PMG value.
0b1	The monitor only measures storage used with the PMG value matching MSMON_CFG_CSU_FLT .PMG.

If MATCH_PMG is 1 and MATCH_PARTID is 0, it is CONSTRAINED UNPREDICTABLE whether the monitor instance:

- Measures the storage used with matching PMG and with any PARTID.
- Measures no storage usage, that is, [MSMON_CSU](#).VALUE is zero.
- Measures the storage used with matching PMG and PARTID, that is, treats MATCH_PARTID as == 1.

MATCH_PARTID, bit [16]

Match PARTID.

Controls whether the monitor measures only storage used with PARTID matching [MSMON_CFG_CSU_FLT](#).PARTID.

MATCH_PARTID	Meaning
0b0	The monitor measures storage used with any PARTID value.
0b1	The monitor only measures storage used with the PARTID value matching MSMON_CFG_CSU_FLT .PARTID.

Bits [15:11]

Reserved, RES0.

OFLOW_LNKG, bits [10:8]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_OFLOW_LNKG == 1:

Overflow linkage event.

Controls signaling of a capture event on overflow of this monitor instance.

OFLOW_LNKG	Meaning
0b000	Overflow of the monitor instance only affects this monitor instance.
0b001	Overflow of this monitor instance signals Capture Event 1.
0b010	Overflow of this monitor instance signals Capture Event 2.
0b011	Overflow of this monitor instance signals Capture Event 3.
0b100	Overflow of this monitor instance signals Capture Event 4.
0b101	Overflow of this monitor instance signals Capture Event 5.
0b110	Overflow of this monitor instance signals Capture Event 6.
0b111	Reserved.

Otherwise:

Reserved, RES0.

TYPE, bits [7:0]

Monitor Type Code. The CSU monitor is TYPE = 0x43.

TYPE is a read-only constant indicating the type of the monitor.

Reads as 0x43.

Access to this field is **RO**.

Accessing MSMON_CFG_CSU_CTL

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MSMON_CFG_CSU_CTL_s must only be accessible from the Secure MPAM feature page.
- MSMON_CFG_CSU_CTL_ns must only be accessible from the Non-secure MPAM feature page.
- MSMON_CFG_CSU_CTL_rt must only be accessible from the Root MPAM feature page.
- MSMON_CFG_CSU_CTL_rl must only be accessible from the Realm MPAM feature page.

MSMON_CFG_CSU_CTL_s, MSMON_CFG_CSU_CTL_ns, MSMON_CFG_CSU_CTL_rt, and MSMON_CFG_CSU_CTL_rl must be separate registers:

- The Secure instance (MSMON_CFG_CSU_CTL_s) accesses the cache storage usage monitor controls used for Secure PARTIDs.
- The Non-secure instance (MSMON_CFG_CSU_CTL_ns) accesses the cache storage usage monitor controls used for Non-secure PARTIDs.
- The Root instance (MSMON_CFG_CSU_CTL_rt) accesses the cache storage usage monitor controls used for Root PARTIDs.
- The Realm instance (MSMON_CFG_CSU_CTL_rl) accesses the cache storage usage monitor controls used for Realm PARTIDs.

When RIS is implemented, loads and stores to MSMON_CFG_CSU_CTL access the cache storage usage monitor configuration settings for the cache resource instance selected by [MSMON_CFG_MON_SEL](#).RIS and the cache storage usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

When RIS is not implemented, loads and stores to MSMON_CFG_CSU_CTL access the cache storage usage monitor configuration settings for the cache storage usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

Accesses to this interface are **RW**.

Accesses to this interface are **RW**.

When FEAT_RME is implemented, accesses to this interface are **RW**.

When FEAT RME is implemented, accesses to this interface are **RW**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

MSMON_CFG_MBWU_CTL, MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register

The MSMON CFG MBWU CTL characteristics are:

Purpose

Controls the MBWU monitor selected by [MSMON CFG MON SEL](#).

MSMON_CFG_MBWU_CTL_s controls the Secure memory bandwidth usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_MBWU_CTL_ns controls Non-secure memory bandwidth usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_MBWU_CTL_rt controls the monitor configuration for the Root PARTID selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_MBWU_CTL_rl controls the monitor configuration for the Realm PARTID selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR](#).HAS_RIS is 1, the monitor instance configuration accessed is for the resource instance currently selected by [MSMON_CFG_MON_SEL](#).RIS and the monitor instance of that resource instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

Configuration

The power domain of MSMON_CFG_MBWU_CTL is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1 and MPAMF_MSMON_IDR.MSMON_MBWU == 1. Otherwise, direct accesses to MSMON_CFG_MBWU_CTL are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CFG_MBWU_CTL is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18
ENCAPT EVNT				CAPT RESET		OFLOW STATUS		OFLOW INTR		OFLOW FRZ		OFLOW CAPT	
SUBTYPE				SCLEN		CEVNT		OFLW		MA			

EN, bit [31]

Enabled.

EN	Meaning
0b0	The monitor instance is disabled and must not collect any information.
0b1	The monitor instance is enabled to collect information according to the configuration of the instance.

CAPT EVNT, bits [30:28]

Capture event selector.

When the selected capture event occurs, [MSMON_MBWU](#) of the monitor instance is copied to [MSMON_MBWU_CAPTURE](#) of the same instance. If the long counter is also implemented, [MSMON_MBWU_L](#) is also copied to [MSMON_MBWU_L_CAPTURE](#).

Select the event that triggers capture from the following:

CAPT_EVNT	Meaning
0b000	No capture event is triggered.
0b001	External capture event 1 (optional, but recommended)
0b010	External capture event 2 (optional)
0b011	External capture event 3 (optional)
0b100	External capture event 4 (optional)
0b101	External capture event 5 (optional)
0b110	External capture event 6 (optional)
0b111	Capture occurs when a MSMON_CAPT_EVNT register in this MSC is written and causes a capture event for the Security state of this monitor. (optional)

The values marked as optional indicate capture event sources that can be omitted in an implementation. Those values representing non-implemented event sources must not trigger a capture event.

When [MPAMF_MBWUMON_IDR.HAS_CAPTURE](#) == 0, access to this field is **RAZ/WI**.

If capture is not implemented for the MBWU monitor type as indicated by [MPAMF_MBWUMON_IDR.HAS_CAPTURE](#) = 0, this field is RAZ/WI.

CAPT_RESET, bit [27]

Reset [MSMON_MBWU](#).VALUE after capture.

Controls whether the VALUE field of the monitor instance is reset to zero immediately after being copied to the corresponding capture register.

CAPT_RESET	Meaning
0b0	MSMON_MBWU .VALUE field of the monitor instance is not reset on capture.
0b1	MSMON_MBWU .VALUE field of the monitor instance is reset on capture.

If capture is not implemented for the MBWU monitor type as indicated by [MPAMF_MBWUMON_IDR.HAS_CAPTURE](#) = 0, this field is RAZ/WI.

This control bit affects both [MSMON_MBWU](#) and [MSMON_MBWU_L](#) in implementations that include [MSMON_MBWU_L](#).

When [MPAMF_MBWUMON_IDR.HAS_CAPTURE](#) == 0, access to this field is **RAZ/WI**.

OFLOW_STATUS, bit [26]

Overflow status.

Indicates whether the value of [MSMON_MBWU](#) has overflowed.

OFLOW_STATUS	Meaning
0b0	MSMON_MBWU .VALUE has not overflowed.
0b1	MSMON_MBWU .VALUE has overflowed at least once since this bit was last written to zero.

If overflow is not possible for an MBWU monitor in the MSC implementation, this field is RAZ/WI.

Overflow status for [MSMON_MBWU_L](#).VALUE is reported in [MSMON_CFG_MBWU_CTL](#).OFLOW_STATUS_L.

If [MPAMF_MBWUMON_IDR.HAS_CEVTN_OFLW](#) is 1 or [MPAMF_MBWUMON_IDR.HAS_OFLOW_LNKG](#) is 1, then a store to [MSMON_MBWU](#) when this field is 1 resets this field to 0.

OFLOW_INTR, bit [25]

Enable interrupt on overflow of [MSMON_MBWU](#).VALUE.

OFLOW_INTR	Meaning
0b0	No interrupt is signaled on an overflow of MSMON_MBWU.VALUE .
0b1	An implementation-specific interrupt is signaled on an overflow of MSMON_MBWU.VALUE .

If overflow is not possible for an MBWU monitor in the MSC implementation, this field is RAZ/WI.

If overflow interrupt is not supported by the MSC implementation, this field is RAZ/WI.

Interrupt enable for overflow of [MSMON_MBWU_L.VALUE](#) is controlled by [MSMON_CFG_MBWU_CTL.OFLOW_INTR_L](#).

When [MSMON_CFG_MBWU_CTL.OFLOW_INTR](#) == 0, access to this field is RAZ/WI.

OFLOW_FRZ, bit [24]

Freeze monitor instance on overflow.

Controls whether [MSMON_MBWU.VALUE](#) field of the monitor instance freezes on an overflow.

OFLOW_FRZ	Meaning
0b0	MSMON_MBWU.VALUE field of the monitor instance wraps on overflow.
0b1	MSMON_MBWU.VALUE field of the monitor instance freezes on overflow. If the increment that caused the overflow was 1, the frozen value is the post-increment value of 0. If the increment that caused the overflow was larger than 1, the frozen value of the monitor might be 0 or a larger value less than the final increment.

If overflow is not possible for the instance of the MBWU monitor in the implementation, this field is RAZ/WI.

When a [MSMON_MBWU.VALUE](#) of a monitor instance is frozen it does not change until [MSMON_CSU](#) register for that instance has been written. If the monitor implements both [MSMON_MBWU](#) and [MSMON_MBWU_L](#) registers, both are frozen. A write to a frozen register unfreezes the count for just that register.

This control bit affects both [MSMON_MBWU](#) and [MSMON_MBWU_L](#) in implementations that include [MSMON_MBWU_L](#).

OFLOW_CAPT, bit [23]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_MBWUMON_IDR.HAS_OFLOW_CAPT == 1:

Capture Monitor on Overflow.

OFLOW_CAPT	Meaning
0b0	Monitor register MSMON_MBWU is not captured on an overflow or when affected by an overflow linkage event.
0b1	Monitor register MSMON_MBWU is captured and the MSMON_MBWU .{NRDY, VALUE} fields are copied to the monitor instance's MSMON_MBWU_CAPTURE register on an overflow or when affected by an overflow linkage event. The monitor instance treats an overflow of this monitor instance as a private capture event. If MSMON_CFG_MBWU_CTL.CEVNT_OFLW is 1, this monitor instance also treats an overflow linkage event as a capture event. If OFLOW_FRZ is 1, the monitor does not continue to count after the overflow or overflow linkage event. If CAPT_RESET is 1, the monitor instance resets to 0.

This bit does not control whether [MSMON_MBWU_L](#) is captured on an overflow or overflow linkage event. See [MSMON_CFG_MBWU_CTL.OFLOW_CAPT_L](#).

Otherwise:

Reserved, RES0.

SUBTYPE, bits [22:20]

Subtype. Type of bandwidth counted by this monitor.

This field is not currently used for MBWU monitors, but reserved for future use.

This field is RAZ/WI.

SCLEN, bit [19]

[MSMON_MBWU.VALUE](#) Scaling Enable.

Enables scaling of [MSMON_MBWU.VALUE](#) by [MPAMF_MBWUMON_IDR.SCALE](#).

SCLEN	Meaning
0b0	MSMON_MBWU.VALUE has bytes counted by the monitor instance.
0b1	MSMON_MBWU.VALUE has bytes counted by the monitor instance, shifted right by MPAMF_MBWUMON_IDR.SCALE .

CEVNT_OFLW, bit [18]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_MBWUMON_IDR.HAS_CEVNT_OFLW == 1:

Capture Event performs overflow behavior.

CEVNT_OFLW	Meaning
0b0	On a capture event matching the CAPT_EVNT field the capture behaviors are performed. The NRDY and VALUE fields are transferred to the monitor instance's capture register.
0b1	On a capture event matching the CAPT_EVNT field the monitor instance treats a capture event as an overflow and the overflow behaviors are performed. The behavior is controlled by the MSMON_CFG_MBWU_CTL.{OFLOW_FRZ, OFLOW_CAPT, OFLOW_CAPT_L, CAPT_RESET} fields. The MSMON_CFG_MBWU_CTL.{OFLOW_STATUS, OFLOW_STATUS_L} fields are set for this monitor instance.

Otherwise:

Reserved, RES0.

MATCH_PMG, bit [17]

Match PMG.

Controls whether the monitor instance only counts data transferred with PMG matching [MSMON_CFG_MBWU_FLT.PMG](#).

MATCH_PMG	Meaning
0b0	The monitor instance counts data transferred with any PMG value.
0b1	The monitor instance only counts data transferred with the PMG value matching MSMON_CFG_MBWU_CTL.PMG .

MATCH_PARTID, bit [16]

Match PARTID.

Controls whether the monitor instance counts only data transferred with PARTID matching [MSMON_CFG_MBWU_CTL.PARTID](#).

MATCH_PARTID	Meaning
0b0	The monitor instance counts data transferred with any PARTID value.
0b1	The monitor instance only counts data transferred with the PARTID value matching MSMON_CFG_MBWU_CTL.PARTID .

OFLOW_STATUS_L, bit [15]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Overflow Status of [MSMON_MBWU_L.VALUE](#) of the monitor instance.

Indicates whether [MSMON_MBWU_L.VALUE](#) has overflowed.

OFLOW_STATUS_L	Meaning
0b0	MSMON_MBWU_L.VALUE has not overflowed.
0b1	MSMON_MBWU_L.VALUE has overflowed at least once since this bit was last written to zero.

If [MPAMF_MBWUMON_IDR.HAS_LONG](#) == 0, this bit is RES0.

Overflow status of [MSMON_MBWU.VALUE](#) is reported in [MSMON_CFG_MBWU_CTL.OFLOW_STATUS](#).

If [MPAMF_MBWUMON_IDR.HAS_CEVTN_OFLW](#) is 1 or [MPAMF_MBWUMON_IDR.HAS_OFLOW_LNKG](#) is 1, then a store to [MSMON_MBWU_L](#) when this field is 1 resets this field to 0.

Otherwise:

Reserved, RES0.

OFLOW_INTR_L, bit [14]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_MBWUMON_IDR.HAS_LONG == 1:

Overflow Interrupt for [MSMON_MBWU_L](#).

Controls whether an MPAM overflow interrupt is generated when [MSMON_MBWU_L.VALUE](#) overflows.

OFLOW_INTR_L	Meaning
0b0	No interrupt is signaled on an overflow of MSMON_MBWU_L.VALUE .
0b1	An implementation-specific interrupt is signaled on overflow of MSMON_MBWU_L.VALUE .

If the overflow interrupt is not supported by the MSC implementation, this field is RAZ/WI.

When [MSMON_CFG_MBWU_CTL.OFLOW_INTR_L](#) overflow == is 0, not accessible to for an MBWU monitor in the MSC implementation, this field is RAZ/WI. **RAZ/WI.**

Otherwise:

Reserved, RES0.

OFLOW_CAPT_L, bit [13]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented), MPAMF_MBWUMON_IDR.HAS_LONG == 1 and MPAMF_MBWUMON_IDR.HAS_OFLOW_CAPT == 1:

Capture Long Monitor on Overflow.

Controls whether [MSMON_MBWU_L](#) is copied to [MSMON_MBWU_L_CAPTURE](#) on an overflow or an overflow linkage event.

OFLOW_CAPT_L	Meaning
0b0	Monitor register MSMON_MBWU_L is not captured on an overflow or when affected by an overflow linkage event.
0b1	Monitor register MSMON_MBWU_L is captured on an overflow or when affected by an overflow linkage event. If OFLOW_FRZ is 1, the monitor does not continue to count after the overflow or overflow linkage event. If CAPT_RESET is 1, the monitor instance resets to 0.

If this bit is 1, this monitor instance treats an overflow of this monitor instance as a private capture event.

If this bit is 1, this monitor instance also treats overflow linkage events for which it qualifies as a private capture event.

Otherwise:

Reserved, RES0.

Bits [12:11]

Reserved, RES0.

OFLOW_LNKG, bits [10:8]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_MBWUMON_IDR.HAS_OFLOW_LNKG == 1:

Overflow linkage event.

Controls signaling of a capture event on overflow of this monitor instance.

OFLOW_LNKG	Meaning
0b000	Overflow of the monitor instance only affects this monitor instance.
0b001	Overflow of this monitor instance signals Capture Event 1.
0b010	Overflow of this monitor instance signals Capture Event 2.
0b011	Overflow of this monitor instance signals Capture Event 3.
0b100	Overflow of this monitor instance signals Capture Event 4.
0b101	Overflow of this monitor instance signals Capture Event 5.
0b110	Overflow of this monitor instance signals Capture Event 6.
0b111	Reserved.

Otherwise:

Reserved, RES0.

TYPE, bits [7:0]

Monitor Type Code. The MBWU monitor is TYPE = 0x42.

TYPE is a read-only constant indicating the type of the monitor.

Reads as 0x42.

Access to this field is **RO**.

Accessing MSMON_CFG_MBWU_CTL

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MSMON_CFG_MBWU_CTL_s must only be accessible from the Secure MPAM feature page.
- MSMON_CFG_MBWU_CTL_ns must only be accessible from the Non-secure MPAM feature page.
- MSMON_CFG_MBWU_CTL_rt must only be accessible from the Root MPAM feature page.
- MSMON_CFG_MBWU_CTL_rl must only be accessible from the Realm MPAM feature page.

MSMON_CFG_MBWU_CTL_s, MSMON_CFG_MBWU_CTL_ns, MSMON_CFG_MBWU_CTL_rt, and MSMON_CFG_MBWU_CTL_rl must be separate registers:

- The Secure instance (MSMON_CFG_MBWU_CTL_s) accesses the memory bandwidth usage monitor controls used for Secure PARTIDs.
- The Non-secure instance (MSMON_CFG_MBWU_CTL_ns) accesses the memory bandwidth usage monitor controls used for Non-secure PARTIDs.
- The Root instance (MSMON_CFG_MBWU_CTL_rt) accesses the memory bandwidth usage monitor controls used for Root PARTIDs.
- The Realm instance (MSMON_CFG_MBWU_CTL_rl) accesses the memory bandwidth usage monitor controls used for Realm PARTIDs.

When RIS is implemented, loads and stores to MSMON_CFG_MBWU_CTL access the monitor configuration settings for the bandwidth resource instance selected by [MSMON_CFG_MON_SEL](#).RIS and the memory bandwidth usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

When RIS is not implemented, loads and stores to MSMON_CFG_MBWU_CTL access the monitor configuration settings for the memory bandwidth usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

MSMON_CFG_MBWU_CTL can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0828	MSMON_CFG_MBWU_CTL_s

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0828	MSMON_CFG_MBWU_CTL_ns

Accesses to this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0828	MSMON_CFG_MBWU_CTL_rt

When FEAT_RME is implemented, accesses to this interface are **RW**.

Component	Frame	Offset	Instance
-----------	-------	--------	----------

MPAM	MPAMF_BASE_r1	0x0828	MSMON_CFG_MBWU_CTL_r1
------	---------------	--------	-----------------------

When FEAT_RME is implemented, accesses to this interface are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

MSMON_CSU_OFSR, MPAM CSU Monitor Overflow Status Register

The MSMON_CSU_OFSR characteristics are:

Purpose

MSMON_CSU_OFSR is a 32-bit read-only register that shows bitmap of CSU monitor instance overflow status for a contiguous group of 32 monitor instances.

MSMON_CSU_OFSR_s gives a bitmap of pending CSU overflow status for 32 Secure CSU monitor instances.
 MSMON_CSU_OFSR_ns gives a bitmap of pending CSU overflow status for 32 Non-secure CSU monitor instances.
 MSMON_CSU_OFSR_rt gives a bitmap of pending CSU overflow status for 32 Root CSU monitor instances.
 MSMON_CSU_OFSR_rl gives a bitmap of pending CSU overflow status for 32 Realm CSU monitor instances.

Configuration

The power domain of MSMON_CSU_OFSR is IMPLEMENTATION DEFINED.

This register is present only when **FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1, MPAMF_MSMON_IDR.MSMON_CSU == 1 and MPAMF_CSUMON_IDR.HAS_OFSR == 1**. Otherwise, direct accesses to MSMON_CSU_OFSR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CSU_OFSR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20
OFPND31	OFPND30	OFPND29	OFPND28	OFPND27	OFPND26	OFPND25	OFPND24	OFPND23	OFPND22	OFPND21	OFPND20

OFPND<i>, bit [i], for i = 31 to 0

Overflow status bitmap for CSU monitor instances. The RIS and the contiguous range of CSU monitor instances are set in [MSMON_CFG_MON_SEL](#). i of 0 corresponds to the CSU monitor instance [MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0](#).

OFPND<i>	Meaning
0b0	CSU monitor instance (MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0 + i) does not have a pending overflow.
0b1	CSU monitor instance (MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0 + i) has a pending overflow.

After reading [MSMON_OFLOW_SR](#) to determine that a CSU monitor instance has a pending overflow and which RIS values have pending overflows, an interrupt service routine could poll groups of 32 monitor instances in a RIS for pending monitors by reading this bitmap and incrementing [MSMON_CFG_MON_SEL.MON_SEL](#) by 32.

Accessing MSMON_CSU_OFSR

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- `MSMON_CSU_OFSR_s` must only be accessible from the Secure MPAM feature page.
- `MSMON_CSU_OFSR_ns` must only be accessible from the Non-secure MPAM feature page.
- `MSMON_CSU_OFSR_rt` must only be accessible from the Root MPAM feature page.
- `MSMON_CSU_OFSR_rl` must only be accessible from the Realm MPAM feature page.

MSMON_CSU_OFSR_s, MSMON_CSU_OFSR_ns, MSMON_CSU_OFSR_rt, and MSMON_CSU_OFSR_rl must be separate registers:

- The Secure instance (MSMON_CSU_OFSR_s) accesses the CSU monitor overflow status bitmap used for Secure PARTIDs.
- The Non-secure instance (MSMON_CSU_OFSR_ns) accesses the CSU monitor overflow status bitmap used for Non-secure PARTIDs.
- The Root instance (MSMON_CSU_OFSR_rt) accesses the CSU monitor overflow status bitmap used for Root PARTIDs.
- The Realm instance (MSMON_CSU_OFSR_rl) accesses the CSU monitor overflow status bitmap used for Realm PARTIDs.

MSMON_CSU_OFSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0858	MSMON_CSU_OFSR_s

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0858	MSMON_CSU_OFSR_ns

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0858	MSMON_CSU_OFSR_rt

When FEAT_RME is implemented, accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0858	MSMON_CSU_OFSR_r1

When FEAT_RME is implemented, accesses to this interface are **RO**.

3005/0907/2022 15:17:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

MSMON_MBWU_OFSR, MPAM MBWU Monitor Overflow Status Register

The MSMON_MBWU_OFSR characteristics are:

Purpose

MSMON_MBWU_OFSR is a 32-bit read-only register that shows bitmap of MBWU monitor instance overflow status for a contiguous group of 32 monitor instances.

MSMON_MBWU_OFSR_s gives a bitmap of pending MBWU overflow status for 32 Secure MBWU monitor instances. MSMON_MBWU_OFSR_ns gives a bitmap of pending MBWU overflow status for 32 Non-secure MBWU monitor instances. MSMON_MBWU_OFSR_rt gives a bitmap of pending MBWU overflow status for 32 Root MBWU monitor instances. MSMON_MBWU_OFSR_rl gives a bitmap of pending MBWU overflow status for 32 Realm MBWU monitor instances.

Configuration

The power domain of MSMON_MBWU_OFSR is IMPLEMENTATION DEFINED.

This register is present only when **FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1, MPAMF_MSMON_IDR.MSMON_MBWU == 1 and MPAMF_MBWUMON_IDR.HAS_OFSR == 1**. Otherwise, direct accesses to MSMON_MBWU_OFSR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_MBWU_OFSR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20
OFPND31	OFPND30	OFPND29	OFPND28	OFPND27	OFPND26	OFPND25	OFPND24	OFPND23	OFPND22	OFPND21	OFPND20

OFPND<i>, bit [i], for i = 31 to 0

Overflow status bitmap for MBWU monitor instances. The RIS and the contiguous range of MBWU monitor instances are set in [MSMON_CFG_MON_SEL](#). i of 0 corresponds to the MBWU monitor instance [MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0](#).

OFPND<i>	Meaning
0b0	MBWU monitor instance (MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0 + i) does not have a pending overflow.
0b1	MBWU monitor instance (MSMON_CFG_MON_SEL.MON_SEL & 0xFFE0 + i) has a pending overflow.

After reading [MSMON_OFLOW_SR](#) to determine that an MBWU monitor instance has a pending overflow and which RIS values have pending overflows, an interrupt service routine could poll groups of 32 monitor instances in a RIS for pending monitors by reading this bitmap and incrementing [MSMON_CFG_MON_SEL.MON_SEL](#) by 32.

A pending overflow may be in either the [MSMON_CFG_MBWU_CTL.OFLOW_STATUS](#) or [MSMON_CFG_MBWU_CTL.OFLOW_STATUS_L](#) field.

Accessing MSMON_MBWU_OFSR

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MSMON_MBWU_OFSR_s must only be accessible from the Secure MPAM feature page.
- MSMON_MBWU_OFSR_ns must only be accessible from the Non-secure MPAM feature page.
- MSMON_MBWU_OFSR_rt must only be accessible from the Root MPAM feature page.
- MSMON_MBWU_OFSR_rl must only be accessible from the Realm MPAM feature page.

MSMON_MBWU_OFSR_s, MSMON_MBWU_OFSR_ns, MSMON_MBWU_OFSR_rt, and MSMON_MBWU_OFSR_rl must be separate registers:

- The Secure instance (MSMON_MBWU_OFSR_s) accesses the MBWU monitor overflow status bitmap used for Secure PARTIDs.
- The Non-secure instance (MSMON_MBWU_OFSR_ns) accesses the MBWU monitor overflow status bitmap used for Non-secure PARTIDs.
- The Root instance (MSMON_MBWU_OFSR_rt) accesses the MBWU monitor overflow status bitmap used for Root PARTIDs.
- The Realm instance (MSMON_MBWU_OFSR_rl) accesses the MBWU monitor overflow status bitmap used for Realm PARTIDs.

MSMON_MBWU_OFSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0898	MSMON_MBWU_OFSR_s

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0898	MSMON_MBWU_OFSR_ns

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0898	MSMON_MBWU_OFSR_rt

When FEAT_RME is implemented, accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0898	MSMON_MBWU_OFSR_rl

When FEAT_RME is implemented, accesses to this interface are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd6d6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_OFLOW_SR, MPAM Monitor Overflow Status Register

The MSMON_OFLOW_SR characteristics are:

Purpose

MSMON_OFLOW_SR is a 32-bit read-only register that shows MPAM monitor overflow status for this MSC.

MSMON_OFLOW_SR_s gives the status of overflows of Secure MPAM monitors. MSMON_OFLOW_SR_ns gives the status of overflows of Non-secure MPAM monitors. MSMON_OFLOW_SR_rt gives the status of overflows of Root MPAM monitors. MSMON_OFLOW_SR_rl gives the status of overflows of Realm MPAM monitors.

Configuration

The power domain of MSMON_OFLOW_SR is IMPLEMENTATION DEFINED.

This register is present only when **FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1 and MPAMF_MSMON_IDR.HAS_OFLOW_SR == 1**. Otherwise, direct accesses to MSMON_OFLOW_SR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_SR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSU_OFLOW_PND	MBWU_OFLOW_PND	RES0														RIS_PND15	RIS_PND14	RIS_PND13	RIS_PND12	RIS_PND11	RIS_PND10	RIS_PND9	RIS_PND8	RIS_PND7	RIS_PND6	RIS_PND5	RIS_PND4	RIS_PND3	RIS_PND2	RIS_PND1	RIS_PND0

CSU_OFLOW_PND, bit [31]

At least one cache storage usage monitor has OFLOW_STATUS of 1 in [MSMON_CFG_CSU_CTL](#).

CSU_OFLOW_PND	Meaning
0b0	There are no cache storage usage monitor instances where MSMON_CFG_CSU_CTL .OFLOW_STATUS is 1.
0b1	MSMON_CFG_CSU_CTL for at least one of the cache storage usage monitor instances has OFLOW_STATUS set to 1.

This field clears when [MSMON_CFG_CSU_CTL](#).OFLOW_STATUS has been reset to 0 for all CSU monitor instances in this MSC.

MBWU_OFLOW_PND, bit [30]

At least one memory bandwidth usage monitor instance has OFLOW_STATUS or OFLOW_STATUS_L of 1 in [MSMON_CFG_MBWU_CTL](#).

MBWU_OFLOW_PND	Meaning
0b0	There are no memory bandwidth usage monitor instances where MSMON_CFG_MBWU_CTL.OFLOW_STATUS is 1.
0b1	MSMON_CFG_MBWU_CTL for at least one of the memory bandwidth usage monitor instances has either OFLOW_STATUS or OFLOW_STATUS_L set to 1.

This field clears when [MSMON_CFG_MBWU_CTL.OFLOW_STATUS](#) and [MSMON_CFG_MBWU_CTL.OFLOW_STATUS_L](#) have been reset to 0 for all MBWU monitor instances in this MSC.

Bits [29:16]

Reserved, RES0.

RIS_PND<r>, bit [r], for r = 15 to 0

Overflow status by RIS.

RIS_PND<r>	Meaning
0b0	RIS r has no unread overflows of any type of monitor.
0b1	RIS r has at least one unread overflow in at least one of the monitor types.

Combined with the CSU_OFLOW_PND and MBWU_OFLOW_PND flags in this register, an interrupt service routine could poll only the monitor types indicated in monitors for the resource instances flagged in this field.

Bit r is set when any monitor instance of any type in resource instance r has OFLOW_STATUS or OFLOW_STATUS_L set to 1.

Accessing MSMON_OFLOW_SR

This register is within the MPAM feature page memory frames.

In a system that supports Secure, Non-secure, Root, and Realm memory maps, there must be MPAM feature pages in all four address maps:

- MSMON_OFLOW_SR_s must only be accessible from the Secure MPAM feature page.
- MSMON_OFLOW_SR_ns must only be accessible from the Non-secure MPAM feature page.
- MSMON_OFLOW_SR_rt must only be accessible from the Root MPAM feature page.
- MSMON_OFLOW_SR_rl must only be accessible from the Realm MPAM feature page.

MSMON_OFLOW_SR_s, MSMON_OFLOW_SR_ns, MSMON_OFLOW_SR_rt, and MSMON_OFLOW_SR_rl must be separate registers:

- The Secure instance (MSMON_OFLOW_SR_s) accesses the monitor overflow status summary of Secure monitors.
- The Non-secure instance (MSMON_OFLOW_SR_ns) accesses the monitor overflow status summary of Non-secure monitors.
- The Root instance (MSMON_OFLOW_SR_rt) accesses the monitor overflow status summary of Root monitors.
- The Realm instance (MSMON_OFLOW_SR_rl) accesses the monitor overflow status summary of Realm monitors.

MSMON_OFLOW_SR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x08F0	MSMON_OFLOW_SR_s

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x08F0	MSMON_OFLOW_SR_ns

Accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08F0	MSMON_OFLOW_SR_rt

When FEAT_RME is implemented, accesses to this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x08F0	MSMON_OFLOW_SR_rl

When FEAT_RME is implemented, accesses to this interface are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBAUTHSTATUS, Authentication Status Register

The TRBAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBAUTHSTATUS are RES0.

[TRBAUTHSTATUS](#) is in the Core power domain.

Attributes

TRBAUTHSTATUS is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	RTNID	RTID						RES0					RLNID	RLID		RES0	SNID	SID	NSNID	NSID											

Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

RTNID	Meaning
0b00	Not implemented.

Access to this field is **RO**.

RTID, bits [25:24]

Root invasive debug.

This field has the same value as [DBGAUTHSTATUS_EL1](#).RTID.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

RLNID	Meaning
0b00	Not implemented.

Access to this field is **RO**.

RLID, bits [13:12]

Realm invasive debug.

This field has the same value as [DBGAUTHSTATUS_EL1](#).RLID.

Bits [11:8]

Reserved, RES0.

SNID, bits [7:6]

Secure non-invasive debug.

SNID	Meaning
0b00	Not implemented.

Access to this field is **RO**.

SID, bits [5:4]

Secure invasive debug.

This field has the same value as [DBGAUTHSTATUS_EL1](#).SID.

NSNID, bits [3:2]

Non-secure non-invasive debug.

NSNID	Meaning
0b00	Not implemented.

Access to this field is **RO**.

NSID, bits [1:0]

Non-secure invasive debug.

This field has the same value as [DBGAUTHSTATUS_EL1](#).NSID.

Accessing TRBAUTHSTATUS

TRBAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFB8	TRBAUTHSTATUS

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

no old file htmldiff from- (new)

TRBBASER_EL1, Trace Buffer Base Address Register

The TRBBASER_EL1 characteristics are:

Purpose

Defines the base address for the trace buffer.

Configuration

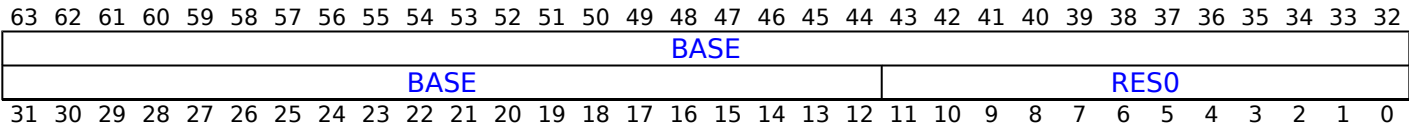
External register TRBBASER_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBBASER_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBBASER_EL1 are RES0.

Attributes

TRBBASER_EL1 is a 64-bit register.

Field descriptions



BASE, bits [63:12]

Trace Buffer Base pointer address. ([TRBBASER_EL1](#).BASE << 12) is the address of the first byte in the trace buffer. Bits [11:0] of the Base pointer address are always zero. If the smallest implemented translation granule is not 4KB, then [TRBBASER_EL1](#)[N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value Log₂(smallest implemented translation granule).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [11:0]

Reserved, RES0.

Accessing TRBBASER_EL1

The PE might ignore a write to [TRBBASER_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBBASER_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

TRBBASER_EL1, Trace Buffer Base Address Register

TRBE	0x000	TRBBASER_EL1
------	-------	--------------

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TRBCIDR0, Component Identification Register 0

The TRBCIDR0 characteristics are:

Purpose

Provides discovery information about the component.
For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBCIDR0 are RES0.

[TRBCIDR0](#) is in the Core power domain.

Attributes

TRBCIDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_0							

Bits [31:8]

Reserved, RES0.

PRMBL_0, bits [7:0]

Component identification preamble, segment 0.
Reads as 0x0D.
Access to this field is **RO**.

Accessing TRBCIDR0

TRBCIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFF0	TRBCIDR0

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

TRBCIDR1, Component Identification Register 1

The TRBCIDR1 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBCIDR1 are RES0.

[TRBCIDR1](#) is in the Core power domain.

Attributes

TRBCIDR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								CLASS		PRMBL_1					

Bits [31:8]

Reserved, RES0.

CLASS, bits [7:4]

Component class.

CLASS	Meaning
0b1001	CoreSight peripheral.

Other values are defined by the CoreSight Architecture.

Access to this field is **RO**.

PRMBL_1, bits [3:0]

Component identification preamble, segment 1.

Reads as 0b0000.

Access to this field is **RO**.

Accessing TRBCIDR1

TRBCIDR1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFF4	TRBCIDR1

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBCIDR2, Component Identification Register 2

The TRBCIDR2 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBCIDR2 are RES0.

[TRBCIDR2](#) is in the Core power domain.

Attributes

TRBCIDR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_2							

Bits [31:8]

Reserved, RES0.

PRMBL_2, bits [7:0]

Component identification preamble, segment 2.

Reads as 0x05.

Access to this field is **RO**.

Accessing TRBCIDR2

TRBCIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFF8	TRBCIDR2

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file	htmldiff from-	(new)
-------------	----------------	-------

no old file

htmldiff from-

(new)

TRBCIDR3, Component Identification Register 3

The TRBCIDR3 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBCIDR3 are RES0.

[TRBCIDR3](#) is in the Core power domain.

Attributes

TRBCIDR3 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_3							

Bits [31:8]

Reserved, RES0.

PRMBL_3, bits [7:0]

Component identification preamble, segment 3.

Reads as 0xB1.

Access to this field is **RO**.

Accessing TRBCIDR3

TRBCIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFFC	TRBCIDR3

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file	htmldiff from-	(new)
-------------	----------------	-------

TRBDEVAFF, Device Affinity Register

The TRBDEVAFF characteristics are:

Purpose

For additional information, see the CoreSight Architecture Specification.

Reads the same value as the [MPIDR_EL1](#) register for the PE that this trace buffer has affinity with.

Depending on the IMPLEMENTATION DEFINED nature of the system, it might be possible that [TRBDEVAFF](#) is read before system firmware has configured the trace buffer and/or the PE or group of PEs that the trace buffer has affinity with. When this is the case, [TRBDEVAFF](#) reads as zero.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVAFF are RES0.

[TRBDEVAFF](#) is in the Core power domain.

Attributes

TRBDEVAFF is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																	MPIDR_EL1														
																	MPIDR_EL1														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MPIDR_EL1, bits [63:0]

Read-only copy of [MPIDR_EL1](#), as seen from the highest implemented Exception level.

Accessing TRBDEVAFF

TRBDEVAFF can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFA8	TRBDEVAFF

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBDEVARCH, Trace Buffer Device Architecture Register

The TRBDEVARCH characteristics are:

Purpose

Provides discovery information for the component.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVARCH are RES0.

[TRBDEVARCH](#) is in the Core power domain.

Attributes

TRBDEVARCH is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHITECT											PRESENT	REVISION			ARCHVER			ARCHPART													

ARCHITECT, bits [31:21]

Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.

ARCHITECT	Meaning
0b01000111011	JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.

Access to this field is **RO**.

PRESENT, bit [20]

DEVARCH present. Defines that [TRBDEVARCH](#) register is present.

PRESENT	Meaning
0b0	Device Architecture information not present.
0b1	Device Architecture information present.

This field reads as 1.

REVISION, bits [19:16]

Revision. Defines the architecture revision of the component.

REVISION	Meaning
0b0000	Revision 0.

All other values are reserved.

Access to this field is **RO**.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component.

ARCHVER	Meaning
0b0000	Trace Buffer Extension version 1.

All other values are reserved.

[TRBDEVARCH](#).ARCHVER and [TRBDEVARCH](#).ARCHPART are also defined as a single field, [TRBDEVARCH](#).ARCHID, so that [TRBDEVARCH](#).ARCHVER is [TRBDEVARCH](#).ARCHID[15:12].

Access to this field is **RO**.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

ARCHPART	Meaning
0xA18	Armv9-A Trace Buffer Extension.

[TRBDEVARCH](#).ARCHVER and [TRBDEVARCH](#).ARCHPART are also defined as a single field, [TRBDEVARCH](#).ARCHID, so that [TRBDEVARCH](#).ARCHPART is [TRBDEVARCH](#).ARCHID[11:0].

Access to this field is **RO**.

Accessing TRBDEVARCH

TRBDEVARCH can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFBC	TRBDEVARCH

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBDEVID, Device Configuration Register

The TRBDEVID characteristics are:

Purpose

Provides discovery information for the component.
For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVID are RES0.
[TRBDEVID](#) is in the Core power domain.

Attributes

TRBDEVID is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																												MPAM			

Bits [31:4]

Reserved, RES0.

MPAM, bits [3:0]

MPAM extensions. Indicates support for Memory Partitioning and Monitoring (MPAM) and the Trace Buffer MPAM extensions.

MPAM	Meaning
0b0000	MPAM not implemented by Trace Buffer Unit.
0b0001	MPAM implemented by Trace Buffer Unit, using default PARTID and PMG values.
0b0010	Trace Buffer MPAM extensions implemented.

When FEAT_MPAM is not implemented by the PE, this field reads as 0b0000.
When FEAT_MPAM is implemented by the PE, the value 0b0000 is not permitted.
FEAT_TRBE_MPAM implements the functionality identified by the value 0b0010.

Accessing TRBDEVID

TRBDEVID can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFC8	TRBDEVID

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TRBDEVID1, Device Configuration Register 1

The TRBDEVID1 characteristics are:

Purpose

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVID1 are RES0.

[TRBDEVID1](#) is in the Core power domain.

Attributes

TRBDEVID1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								PMG_MAX								PARTID_MAX															

Bits [31:24]

Reserved, RES0.

PMG_MAX, bits [23:16] When FEAT_TRBE_MPAM is implemented:

Largest permitted PMG value. The [TRBMPAM](#).PMG field must implement at least enough bits to represent [TRBDEVID1](#).PMG_MAX.

Otherwise:

Reserved, RES0.

PARTID_MAX, bits [15:0] When FEAT_TRBE_MPAM is implemented:

Largest permitted PARTID value. The [TRBMPAM](#).PARTID field must implement at least enough bits to represent [TRBDEVID1](#).PARTID_MAX.

Otherwise:

Reserved, RES0.

Accessing TRBDEVID1

TRBDEVID1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFC4	TRBDEVID1

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file htmldiff from- (new)

TRBDEVID2, Device Configuration Register 2

The TRBDEVID2 characteristics are:

Purpose

Provides discovery information for the component.
For additional information, see the CoreSight Architecture Specification.

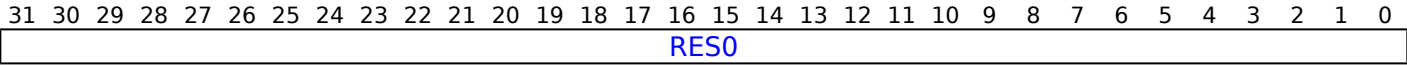
Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVID2 are RES0.
[TRBDEVID2](#) is in the Core power domain.

Attributes

TRBDEVID2 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRBDEVID2

TRBDEVID2 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFC0	TRBDEVID2

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

TRBDEVTYPE, Device Type Register

The TRBDEVTYPE characteristics are:

Purpose

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by [TRBDEVTYPE](#) about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBDEVTYPE are RES0.

[TRBDEVTYPE](#) is in the Core power domain.

Attributes

TRBDEVTYPE is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0												SUB				MAJOR			

Bits [31:8]

Reserved, RES0.

SUB, bits [7:4]

Component sub-type.

SUB	Meaning
0b0010	When MAJOR == 0x1 (Trace sink): Trace buffer or router.

This field reads as 0x2.

MAJOR, bits [3:0]

Component major type.

MAJOR	Meaning
0b0001	Trace sink.

Other values are defined by the CoreSight Architecture.

Access to this field is **RO**.

Accessing TRBDEVTYPE

TRBDEVTYPE can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFCC	TRBDEVTYPE

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBIDR_EL1, Trace Buffer ID Register

The TRBIDR_EL1 characteristics are:

Purpose

Describes constraints on using the Trace Buffer Unit to an external debugger.

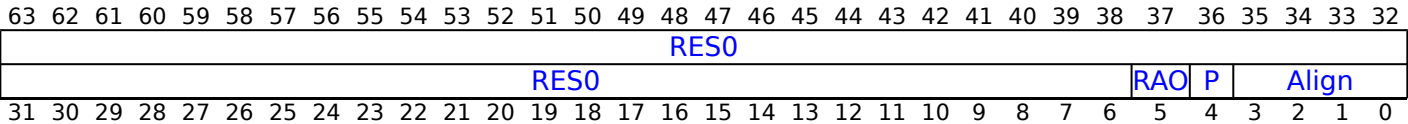
Configuration

This register is present only when FEAT_TRBE is implemented. Otherwise, direct accesses to TRBIDR_EL1 are RES0.

Attributes

TRBIDR_EL1 is a 64-bit register.

Field descriptions



Bits [63:6]

Reserved, RES0.

Bit [5]

Reserved, RAO.

P, bit [4]

This field reads as an UNKNOWN value.

Align, bits [3:0]

Defines the minimum alignment constraint for writes to [TRBPTR_EL1](#) and [TRBTRG_EL1](#).

Align	Meaning
0b0000	Byte.
0b0001	Halfword.
0b0010	Word.
0b0011	Doubleword.
0b0100	16 bytes.
0b0101	32 bytes.
0b0110	64 bytes.
0b0111	128 bytes.
0b1000	256 bytes.
0b1001	512 bytes.
0b1010	1KB.
0b1011	2KB.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBIDR_EL1

TRBIDR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x030	TRBIDR_EL1

Accesses to this interface are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBITCTRL, Integration Mode Control Register

The TRBITCTRL characteristics are:

Purpose

A component can use [TRBITCTRL](#) to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBITCTRL are RES0.

[TRBITCTRL](#) is in the Core power domain.

Attributes

TRBITCTRL is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															IME

Bits [31:1]

Reserved, RES0.

IME, bit [0]

When topology detection or integration functionality is implemented:

Integration Mode Enable.

IME	Meaning
0b0	Component functional mode.
0b1	Component integration mode. Support for topology detection and integration testing is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRBITCTRL

The PE might ignore a write to TRBITCTRL if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBITCTRL can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xF00	TRBITCTRL

This interface is accessible as follows:

- When DoubleLockStatus(), or !IsCorePowered() or !AllowExternalTraceAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TRBLAR, Lock Access Register

The TRBLAR characteristics are:

Purpose

For components that implement the Software Lock, used to lock and unlock the Software Lock. This component does not implement the Software Lock.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBLAR are RES0.

[TRBLAR](#) is in the Core power domain.

Attributes

TRBLAR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																WI															

Bits [31:0]

Reserved, WI.

Software Lock Key. The Software Lock is not implemented.

This field ignores writes.

Accessing TRBLAR

TRBLAR can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFB0	TRBLAR

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **WO**.

no old file

htmldiff from-

(new)

TRBLIMITR_EL1, Trace Buffer Limit Address Register

The TRBLIMITR_EL1 characteristics are:

Purpose

Defines the top address for the trace buffer, and controls the trace buffer modes and enable.

Configuration

External register TRBLIMITR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBLIMITR_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBLIMITR_EL1 are RES0.

Attributes

TRBLIMITR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32						
																LIMIT																					
LIMIT												RES0										XE		nVM		TM		FM		E							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

LIMIT, bits [63:12]

Trace buffer Limit pointer address. ([TRBLIMITR_EL1](#).LIMIT << 12) is the address of the last byte in the trace buffer plus one. Bits [11:0] of the Limit pointer address are always zero. If the smallest implemented translation granule is not 4KB, then [TRBLIMITR_EL1](#)[N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value Log₂(smallest implemented translation granule).

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [11:7]

Reserved, RES0.

XE, bit [6]

Trace Buffer Unit External mode enable. Controls whether the Trace Buffer Unit is enabled when SelfHostedTraceEnabled() == FALSE.

XE	Meaning
0b0	Trace Buffer Unit is not enabled by this control.
0b1	If SelfHostedTraceEnabled() is FALSE, the Trace Buffer Unit is enabled.

If SelfHostedTraceEnabled() == TRUE, then [TRBLIMITR_EL1](#).E controls whether the Trace Buffer Unit is enabled.

All output is discarded by the Trace Buffer Unit when the Trace Buffer Unit is disabled.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

nVM, bit [5]

Address mode.

nVM	Meaning
0b0	The trace buffer pointers are virtual addresses.
0b1	The trace buffer pointers are: <ul style="list-style-type: none"> • Physical address in the owning security state if the owning translation regime has no stage 2 translation. • Intermediate physical addresses in the owning security state if the owning translation regime has stage 2 translations.

When SelfHostedTraceEnabled() == FALSE, the trace buffer pointers are always physical addresses.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When !SelfHostedTraceEnabled(), access to this field is **RES1**.
- Otherwise, access to this field is **RW**.

TM, bits [4:3]

Trigger mode.

TM	Meaning
0b00	Stop on trigger. Flush trace, then stop collection and set TRBSR_EL1 .IRQ to 1 on Trigger Event.
0b01	IRQ on trigger. Continue collection and set TRBSR_EL1 .IRQ to 1 on Trigger Event.
0b11	Ignore trigger. Continue collection and leave TRBSR_EL1 .IRQ unchanged on Trigger Event.

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

FM, bits [2:1]

Trace buffer mode.

FM	Meaning
0b00	Fill mode. Stop collection and set TRBSR_EL1 .IRQ to 1 on current write pointer wrap.
0b01	Wrap mode. Continue collection and set TRBSR_EL1 .IRQ to 1 on current write pointer wrap.
0b11	Circular Buffer mode. Continue collection and leave TRBSR_EL1 .IRQ unchanged on current write pointer wrap.

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

E, bit [0]

Trace Buffer Unit enable. Controls whether the Trace Buffer Unit is enabled when SelfHostedTraceEnabled() == TRUE.

E	Meaning
0b0	Trace Buffer Unit is not enabled by this control.
0b1	If SelfHostedTraceEnabled() is TRUE, the Trace Buffer Unit is enabled.

If FEAT_TRBE_EXT is implemented and SelfHostedTraceEnabled() == FALSE, then [TRBLIMITR_EL1.XE](#) controls whether the Trace Buffer Unit is enabled.

If FEAT_TRBE_EXT is not implemented, then the Trace Buffer Unit is disabled when SelfHostedTraceEnabled() == FALSE.

All output is discarded by the Trace Buffer Unit when the Trace Buffer Unit is disabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing TRBLIMITR_EL1

The PE might ignore a write to [TRBLIMITR_EL1](#), other than a write that modifies [TRBLIMITR_EL1.E](#) or [TRBLIMITR_EL1.XE](#) as appropriate, if any of the following apply:

- [TRBLIMITR_EL1.E](#) == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1.XE](#) == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBLIMITR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x010	TRBLIMITR_EL1

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBLSR, Lock Status Register

The TRBLSR characteristics are:

Purpose

Indicates the Software Lock is not implemented.
For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBLSR are RES0.
[TRBLSR](#) is in the Core power domain.

Attributes

TRBLSR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																RAZ		SLI													

Bits [31:3]

Reserved, RES0.

Bits [2:1]

Reserved, RAZ.
Not thirty-two bit. Describes the size of the [TRBLAR](#) register.
This field reads-as-zero.

SLI, bit [0]

Indicates the Software Lock is not implemented.

SLI	Meaning
0b0	Software Lock is not implemented. Writes to the TRBLAR are ignored.
0b1	Software Lock is implemented.

Access to this field is **RAZ/WI**.

Accessing TRBLSR

TRBLSR can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

TRBLSR, Lock Status Register

TRBE	0xFB4	TRBLSR
------	-------	--------

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6ffffbdbc66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBMAR_EL1, Trace Buffer Memory Attribute Register

The TRBMAR_EL1 characteristics are:

Purpose

Controls Trace Buffer Unit accesses to memory.

Configuration

External register TRBMAR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBMAR_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBMAR_EL1 are RES0.

Attributes

TRBMAR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0												PAS		SH		Attr															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:12]

Reserved, RES0.

PAS, bits [11:10]

When EL3 is implemented:

Physical address specifier. Defines the PAS attribute for memory addressed by the buffer.

PAS	Meaning	Applies when
0b00	Secure.	
0b01	Non-secure.	
0b10	Root.	When FEAT_RME is implemented
0b11	Realm.	When FEAT_RME is implemented

All other values are reserved.

If the Trace Buffer Unit is using external mode and the IMPLEMENTATION DEFINED authentication interface prohibits invasive debug of the Security state corresponding to the physical address space selected by [TRBMAR_EL1](#).PAS, then when the Trace Buffer Unit receives trace data from the trace unit, it does not write the trace data to memory and generates a trace buffer management event. That is, if any of the following apply:

- ExternalInvasiveDebugEnabled() == FALSE.
- ExternalSecureInvasiveDebugEnabled() == FALSE and [TRBMAR_EL1](#).PAS is 0b00.
- FEAT_RME is implemented, ExternalRootInvasiveDebugEnabled() == FALSE, and [TRBMAR_EL1](#).PAS is 0b10.
- FEAT_RME is implemented, ExternalRealmInvasiveDebugEnabled() == FALSE, and [TRBMAR_EL1](#).PAS is 0b11.

This field is ignored by the PE when SelfHostedTraceEnabled() == TRUE.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SH, bits [9:8]

Trace buffer shareability domain. Defines the shareability domain for Normal memory used by the trace buffer.

SH	Meaning
0b00	Non-shareable.
0b10	Outer Shareable.
0b11	Inner Shareable.

All other values are reserved.

This field is ignored when [TRBMAR_EL1.Attr](#) specifies any of the following memory types:

- Any Device memory type.
- Normal memory, Inner Non-cacheable, Outer Non-cacheable.

All Device and Normal Inner Non-cacheable Outer Non-cacheable memory regions are always treated as Outer Shareable.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Attr, bits [7:0]

When TRBMAR_EL1.Attr == 0bxxxx0000:

Trace buffer memory type and attributes. Defines the memory type and, for Normal memory, the cacheability attributes, for memory addressed by the trace buffer.

Attr	Meaning	Applies when
0x00	Device-nGnRnE memory.	
0x40	Normal memory, Inner Non-cacheable, Outer Non-cacheable with the XS attribute set to 0.	When FEAT_XS is implemented
0xA0	Normal memory, Inner Write-through Cacheable, Outer Write-through Cacheable, Non-transient, Read-Allocate with the XS attribute set to 0.	When FEAT_XS is implemented
0xF0	Tagged Normal memory, Outer Write-Back Non-transient, Read-allocate Write-allocate.	When FEAT_MTE2 is implemented

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

When TRBMAR_EL1.Attr == 0b0000xxxx and TRBMAR_EL1.Attr != 0b00000000:

Trace buffer memory attributes. Defines the Device memory attributes for memory addressed by the trace buffer.

Attr	Meaning	Applies when
0x04	Device-nGnRE memory.	
0x08	Device-nGRE memory.	
0x0C	Device-GRE memory.	
0x01	Device-nGnRnE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x05	Device-nGnRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x09	Device-nGRE memory with the XS attribute set to 0.	When FEAT_XS is implemented
0x0D	Device-GRE memory with the XS attribute set to 0.	When FEAT_XS is implemented

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

When TRBMAR_EL1.Attr != 0bxxxx0000 and TRBMAR_EL1.Attr != 0b0000xxxx:

Trace buffer memory type and attributes. Defines the memory type and, for Normal memory, the Outer and Inner cacheability attributes, for memory addressed by the trace buffer.

Attr	Meaning
0b0001xxxx	Normal memory, Outer Write-Through Transient, Write-allocate.
0b0010xxxx	Normal memory, Outer Write-Through Transient, Read-allocate.
0b0011xxxx	Normal memory, Outer Write-Through Transient, Read-allocate Write-allocate.
0b0100xxxx	Normal memory, Outer Non-cacheable.
0b0101xxxx	Normal memory, Outer Write-Back Transient, Write-allocate.
0b0110xxxx	Normal memory, Outer Write-Back Transient, Read-allocate.
0b0111xxxx	Normal memory, Outer Write-Back Transient, Read-allocate Write-allocate.
0b1000xxxx	Normal memory, Outer Write-Through Non-transient, No allocate.
0b1001xxxx	Normal memory, Outer Write-Through Non-transient, Write-allocate.
0b1010xxxx	Normal memory, Outer Write-Through Non-transient, Read-allocate.
0b1011xxxx	Normal memory, Outer Write-Through Non-transient, Read-allocate Write-allocate.
0b1100xxxx	Normal memory, Outer Write-Back Non-transient, No allocate.
0b1101xxxx	Normal memory, Outer Write-Back Non-transient, Write-allocate.
0b1110xxxx	Normal memory, Outer Write-Back Non-transient, Read-allocate.
0b1111xxxx	Normal memory, Outer Write-Back Non-transient, Read-allocate Write-allocate.
0bxxxx0001	Normal memory, Inner Write-Through Transient, Write-allocate.
0bxxxx0010	Normal memory, Inner Write-Through Transient, Read-allocate.
0bxxxx0011	Normal memory, Inner Write-Through Transient, Read-allocate Write-allocate.
0bxxxx0100	Normal memory, Inner Non-cacheable.
0bxxxx0101	Normal memory, Inner Write-Back Transient, Write-allocate.
0bxxxx0110	Normal memory, Inner Write-Back Transient, Read-allocate.
0bxxxx0111	Normal memory, Inner Write-Back Transient, Read-allocate Write-allocate.
0bxxxx1000	Normal memory, Inner Write-Through Non-transient, No allocate.
0bxxxx1001	Normal memory, Inner Write-Through Non-transient, Write-allocate.
0bxxxx1010	Normal memory, Inner Write-Through Non-transient, Read-allocate.
0bxxxx1011	Normal memory, Inner Write-Through Non-transient, Read-allocate Write-allocate.
0bxxxx1100	Normal memory, Inner Write-Back Non-transient, No allocate.
0bxxxx1101	Normal memory, Inner Write-Back Non-transient, Write-allocate.
0bxxxx1110	Normal memory, Inner Write-Back Non-transient, Read-allocate.
0bxxxx1111	Normal memory, Inner Write-Back Non-transient, Read-allocate Write-allocate.

The reset behavior of this field is:

- On a Cold reset, when FEAT_TRBE_EXT is implemented, this field resets to an architecturally UNKNOWN value.
- On a Warm reset, when FEAT_TRBE_EXT is not implemented, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRBMAR_EL1

The PE might ignore a write to [TRBMAR_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBMAR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x028	TRBMAR_EL1

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBMPAM, Trace Buffer MPAM Configuration Register

The TRBMPAM characteristics are:

Purpose

Defines the PARTID, PMG, and MPAM_SP values used by the trace buffer unit.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBMPAM are RES0.

Attributes

TRBMPAM is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
RES0																																			
RES0								EN		MPAM_SP				PMG								PARTID													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:27]

Reserved, RES0.

EN, bit [26]

Enable. Enables use of non-default MPAM values.

EN	Meaning
0b0	Use default MPAM values.
0b1	Use TRBMPAM .{PARTID, PMG, MPAM_SP}.

This field is ignored by the PE when SelfHostedTraceEnabled() == TRUE.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

MPAM_SP, bits [25:24]

Partition Identifier space. Selects the PARTID space.

MPAM_SP	Meaning	Applies when
0b00	PARTID is in the Secure PARTID space.	
0b01	PARTID is in the Non-secure PARTID space.	
0b10	PARTID is in the Root PARTID space.	When FEAT_RME is implemented
0b11	PARTID is in the Realm PARTID space.	When FEAT_RME is implemented

All other values are reserved.

If `SecureNoninvasiveDebugEnabled() == FALSE`, the value `0b00` is treated as `0b01`.

If `FEAT_RME` is implemented and `RootNoninvasiveDebugEnabled() == FALSE`, the value `0b10` is treated as `0b01`.

If `FEAT_RME` is implemented and `RealmNoninvasiveDebugEnabled() == FALSE`, the value `0b11` is treated as `0b01`.

This field is ignored by the PE when `SelfHostedTraceEnabled() == TRUE`.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PMG, bits [23:16]

Performance Monitoring Group. Selects the PMG.

Only sufficient low-order bits are required to represent the [TRBDEVID1](#).PMG_MAX. Higher-order bits are RES0.

This field is ignored by the PE when `SelfHostedTraceEnabled() == TRUE`.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PARTID, bits [15:0]

Partition Identifier. Selects the PARTID.

Only sufficient low-order bits are required to represent the [TRBDEVID1](#).PARTID_MAX. Higher-order bits are RES0.

This field is ignored by the PE when `SelfHostedTraceEnabled() == TRUE`.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing TRBMPAM

TRBMPAM can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x040	TRBMPAM

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR0, Peripheral Identification Register 0

The TRBPIDR0 characteristics are:

Purpose

Provides discovery information about the component.
For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR0 are RES0.
[TRBPIDR0](#) is in the Core power domain.

Attributes

TRBPIDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PART_0							

Bits [31:8]

Reserved, RES0.

PART_0, bits [7:0]

Part number, bits [7:0].
The part number is selected by the designer of the component, and is stored in [TRBPIDR1](#).PART_1 and [TRBPIDR0](#).PART_0.
This field has an IMPLEMENTATION DEFINED value.
Access to this field is **RO**.

Accessing TRBPIDR0

TRBPIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFE0	TRBPIDR0

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR1, Peripheral Identification Register 1

The TRBPIDR1 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR1 are RES0.

[TRBPIDR1](#) is in the Core power domain.

Attributes

TRBPIDR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								DES 0				PART 1			

Bits [31:8]

Reserved, RES0.

DES_0, bits [7:4]

Designer, JEP106 identification code, bits [3:0]. [TRBPIDR1](#).DES_0 and [TRBPIDR2](#).DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 identification code is 0x3B.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PART_1, bits [3:0]

Part number, bits [11:8].

The part number is selected by the designer of the component, and is stored in [TRBPIDR1](#).PART_1 and [TRBPIDR0](#).PART_0.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBPIDR1

TRBPIDR1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFE4	TRBPIDR1

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR2, Peripheral Identification Register 2

The TRBPIDR2 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR2 are RES0.

[TRBPIDR2](#) is in the Core power domain.

Attributes

TRBPIDR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																REVISION		JEDEC		DES 1											

Bits [31:8]

Reserved, RES0.

REVISION, bits [7:4]

Component major revision. [TRBPIDR2.REVISION](#) and [TRBPIDR3.REVAND](#) together form the revision number of the component, with [TRBPIDR2.REVISION](#) being the most significant part and [TRBPIDR3.REVAND](#) the least significant part. When a component is changed, [TRBPIDR2.REVISION](#) or [TRBPIDR3.REVAND](#) are increased to ensure that software can differentiate the different revisions of the component. [TRBPIDR3.REVAND](#) should be set to 0b0000 when [TRBPIDR2.REVISION](#) is increased.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

JEDEC, bit [3]

JEDEC-assigned JEP106 implementer code is used.

Reads as 0b1.

Access to this field is **RO**.

DES_1, bits [2:0]

Designer, JEP106 identification code, bits [6:4]. [TRBPIDR1.DES_0](#) and [TRBPIDR2.DES_1](#) together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be the same as the

implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 identification code is 0x3B.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBPIDR2

TRBPIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFE8	TRBPIDR2

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR3, Peripheral Identification Register 3

The TRBPIDR3 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR3 are RES0.

[TRBPIDR3](#) is in the Core power domain.

Attributes

TRBPIDR3 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
												RES0															REVAND			CMOD		

Bits [31:8]

Reserved, RES0.

REVAND, bits [7:4]

Component minor revision. [TRBPIDR2](#).REVISION and [TRBPIDR3](#).REVAND together form the revision number of the component, with [TRBPIDR2](#).REVISION being the most significant part and [TRBPIDR3](#).REVAND the least significant part. When a component is changed, [TRBPIDR2](#).REVISION or [TRBPIDR3](#).REVAND are increased to ensure that software can differentiate the different revisions of the component. [TRBPIDR3](#).REVAND should be set to 0b0000 when [TRBPIDR2](#).REVISION is increased.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

CMOD, bits [3:0]

Customer Modified.

Indicates the component has been modified.

A value of 0b0000 means the component is not modified from the original design.

Any other value means the component has been modified in an IMPLEMENTATION DEFINED way.

For any two components with the same Unique Component Identifier:

- If the value of the CMOD fields of both components is zero then the components are identical.

- If the CMOD fields of both components have the same nonzero value then this does not necessarily mean that they have the same modifications.
- If the value of the CMOD field of either of the two components is nonzero, they might not be identical, even though they have the same Unique Component Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBPIDR3

TRBPIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFEC	TRBPIDR3

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR4, Peripheral Identification Register 4

The TRBPIDR4 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR4 are RES0.

[TRBPIDR4](#) is in the Core power domain.

Attributes

TRBPIDR4 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								SIZE			DES 2				

Bits [31:8]

Reserved, RES0.

SIZE, bits [7:4]

Size of the component.

The distance from the start of the address space used by this component to the end of the component identification registers.

A value of 0b0000 means one of the following is true:

- The component uses a single 4KB block.
- The component uses an IMPLEMENTATION DEFINED number of 4KB blocks.

Any other value means the component occupies $2^{\text{TRBPIDR4.SIZE}}$ 4KB blocks.

Using this field to indicate the size of the component is deprecated. This field might not correctly indicate the size of the component. Arm recommends that software determine the size of the component from the Unique Component Identifier fields, and other IMPLEMENTATION DEFINED registers in the component.

Reads as 0b0000.

Access to this field is **RO**.

DES_2, bits [3:0]

Designer, JEP106 continuation code. This is the JEDEC-assigned JEP106 bank identifier for the designer of the component, minus 1. The code identifies the designer of the component, which might not be the same as the

implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 bank is 5, meaning this field has the value 0x4.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRBPIDR4

TRBPIDR4 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFD0	TRBPIDR4

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

TRBPIDR5, Peripheral Identification Register 5

The TRBPIDR5 characteristics are:

Purpose

Provides discovery information about the component.
For additional information, see the CoreSight Architecture Specification.

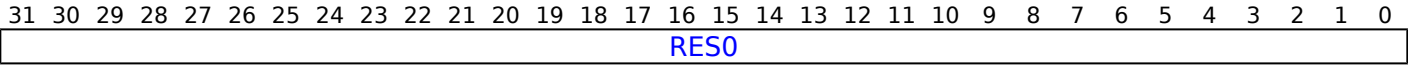
Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR5 are RES0.
[TRBPIDR5](#) is in the Core power domain.

Attributes

TRBPIDR5 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRBPIDR5

TRBPIDR5 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFD4	TRBPIDR5

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

TRBPIDR6, Peripheral Identification Register 6

The TRBPIDR6 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

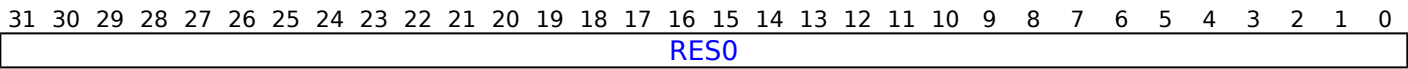
This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR6 are RES0.

[TRBPIDR6](#) is in the Core power domain.

Attributes

TRBPIDR6 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRBPIDR6

TRBPIDR6 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFD8	TRBPIDR6

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPIDR7, Peripheral Identification Register 7

The TRBPIDR7 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPIDR7 are RES0.

[TRBPIDR7](#) is in the Core power domain.

Attributes

TRBPIDR7 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																RES0															

Bits [31:0]

Reserved, RES0.

Accessing TRBPIDR7

TRBPIDR7 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0xFDC	TRBPIDR7

This interface is accessible as follows:

- When DoubleLockStatus() or !IsCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBPTR_EL1, Trace Buffer Write Pointer Register

The TRBPTR_EL1 characteristics are:

Purpose

Defines the current write pointer for the trace buffer.

Configuration

External register TRBPTR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBPTR_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBPTR_EL1 are RES0.

Attributes

TRBPTR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																PTR															
																PTR															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PTR, bits [63:0]

Trace Buffer current write pointer address.

Defines the virtual address of the next entry to be written to the trace buffer.

The architecture places restrictions on the values that software can write to the pointer.

Note

As a result of the restrictions an implementation might treat some of PTR[M:0] as RES0, where M is defined by [TRBIDR_EL1](#).Align.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing TRBPTR_EL1

The PE might ignore a write to [TRBPTR_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBPTR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x008	TRBPTR_EL1

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

TRBSR_EL1, Trace Buffer Status/syndrome Register

The TRBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software for a trace buffer management event.

Configuration

External register TRBSR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBSR_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBSR_EL1 are RES0.

Attributes

TRBSR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
RES0								MSS2																								
EC						RES0		DAT	IRQ	TRG	WRAP	RES0	EA	S	RES0	MSS																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:56]

Reserved, RES0.

MSS2, bits [55:32]

Management event Specific Syndrome 2. Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

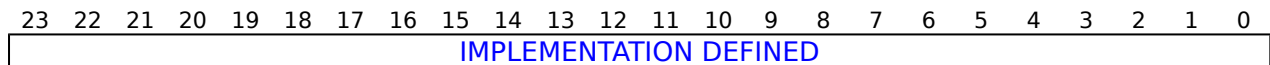
MSS2 encoding for other trace buffer management events

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																							

Bits [23:0]

Reserved, RES0.

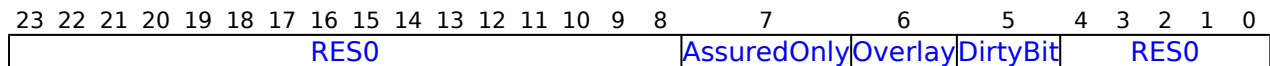
MSS2 encoding for a buffer management event for an IMPLEMENTATION DEFINED reason



IMPLEMENTATION DEFINED, bits [23:0]

IMPLEMENTATION DEFINED.

MSS2 encoding for stage 1 or stage 2 Data Aborts on write to trace buffer



Bits [23:8]

Reserved, RES0.

AssuredOnly, bit [7]

When FEAT_THE is implemented, TRBSR_EL1.EC == 0b100101 and GetTRBSR_EL1_FSC() == 0b0011xx:

AssuredOnly flag. If a memory access generates a stage 2 Data Abort, then this field holds information about the fault.

AssuredOnly	Meaning
0b0	Data Abort is not due to AssuredOnly.
0b1	Data Abort is due to AssuredOnly.

Otherwise:

Reserved, RES0.

Overlay, bit [6]

When (FEAT_S1POE is implemented or FEAT_S2POE is implemented) and GetTRBSR_EL1_FSC() == 0b0011xx:

Overlay flag. If a memory access generates a Data Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Data Abort due to Base Permissions.
0b1	Data Abort due to Overlay Permissions.

Otherwise:

Reserved, RES0.

DirtyBit, bit [5]

When (FEAT_S1PIE is implemented or FEAT_S2PIE is implemented) and GetTRBSR_EL1_FSC() == 0b0011xx:

DirtyBit flag. If a memory access generates a Data Abort (Write Access) for a Permission fault (When using Indirect Permission), then this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to state of nDirty / Dirty bit.
0b1	Permission Fault is due to state of nDirty / Dirty bit.

Otherwise:

Reserved, RES0.

Bits [4:0]

Reserved, RES0.

EC, bits [31:26]

Event Class. Top-level description of the cause of the trace buffer management event.

EC	Meaning	MSS	Applies when
0b000000	Other trace buffer management event. All trace buffer management events other than those described by the other defined event class codes.	MSS encoding for other trace buffer management events	
0b011110	Granule Protection Check fault on write to trace buffer, other than Granule Protection Fault (GPF). That is, any of the following: <ul style="list-style-type: none"> Granule Protection Table (GPT) address size fault. GPT walk fault. Synchronous External abort on GPT fetch. A GPF on translation table walk or update is reported as either a Stage 1 or Stage 2 Data Abort, as appropriate. Other GPFs are reported as a Stage 1 Data Abort.	MSS encoding for other trace buffer management events	When FEAT_RME is implemented
0b011111	Buffer management event for an IMPLEMENTATION DEFINED reason.	MSS encoding for a buffer management event for an IMPLEMENTATION DEFINED reason	
0b100100	Stage 1 Data Abort on write to trace buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer	
0b100101	Stage 2 Data Abort on write to trace buffer.	MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer	

All other values are reserved.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bits [25:24]

Reserved, RES0.

DAT, bit [23]

Data. Indicates when the Trace Buffer Unit has trace data that has not yet been written to memory.

DAT	Meaning
0b0	Internal buffers are empty. All Trace operations Accepted by the Trace Buffer Unit will Complete in finite time.
0b1	Internal buffers are not empty.

When [TRBSR_EL1](#).{DAT, S} is {0, 1}, meaning Collection is stopped and the Trace Buffer Unit internal buffers are empty, then all trace data has been written to memory. An additional Data Synchronization Barrier may be required to ensure that the writes are Complete. When [TRBSR_EL1](#).DAT is 0 and Collection is not stopped, there may still be trace data held by the trace unit that the Trace Buffer Unit has not Accepted.

That is, [TRBSR_EL1](#).DAT reads as 1 when the Trace Buffer Unit has Accepted trace data from the trace unit, but has not yet written it to memory.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

IRQ, bit [22]

Maintenance interrupt status.

IRQ	Meaning
0b0	Maintenance interrupt is not asserted.
0b1	Maintenance interrupt is asserted.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

TRG, bit [21]

Triggered.

TRG	Meaning
0b0	No Detected Trigger has been observed since this field was last cleared to zero.
0b1	A Detected Trigger has been observed since this field was last cleared to zero.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

WRAP, bit [20]

Wrapped.

WRAP	Meaning
0b0	The current write pointer has not wrapped since this field was last cleared to zero.
0b1	The current write pointer has wrapped since this field was last cleared to zero.

For each byte of trace the Trace Buffer Unit Accepts and writes to the trace buffer at the address in the current write pointer; if the current write pointer is equal to the Limit pointer minus one, the current write pointer is wrapped by setting it to the Base pointer, and this field is set to 1.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bit [19]

Reserved, RES0.

EA, bit [18]

When the PE sets this bit as the result of an External Abort:

External Abort.

EA	Meaning
0b0	An External Abort has not been asserted.
0b1	An External Abort has been asserted and detected by the Trace Buffer Unit.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S, bit [17]

Stopped.

S	Meaning
0b0	Collection has not been stopped.
0b1	Collection is stopped.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Bit [16]

Reserved, RES0.

MSS, bits [15:0]

Management event Specific Syndrome. Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

MSS encoding for other trace buffer management events

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										BSC					

Bits [15:6]

Reserved, RES0.

BSC, bits [5:0]

Trace buffer status code.

BSC	Meaning
0b000000	Collection not stopped, or access not allowed.
0b000001	Trace buffer filled. Collection stopped because the current write pointer wrapped to the base pointer and the trace buffer mode is Fill mode.
0b000010	Trigger Event. Collection stopped because of a Trigger Event. See TRBTRG_EL1 for more information.
0b000011	Manual Stop. Collection stopped because of a Manual Stop event. See TRBCR .ManStop for more information.

All other values are reserved.

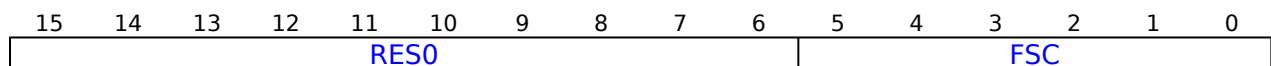
MSS encoding for a buffer management event for an IMPLEMENTATION DEFINED reason



IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED.

MSS encoding for stage 1 or stage 2 Data Aborts on write to trace buffer



Bits [15:6]

Reserved, RES0.

FSC, bits [5:0]

Fault Status Code.

FSC	Meaning	Applies when
0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Asynchronous External abort.	
0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented

0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

Accessing TRBSR_EL1

The PE might ignore a write to [TRBSR_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBSR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x018	TRBSR_EL1

Accesses to this interface are **RW**.

no old file

htmldiff from-

(new)

TRBTRG_EL1, Trace Buffer Trigger Counter Register

The TRBTRG_EL1 characteristics are:

Purpose

Specifies the number of bytes of trace to capture following a Detected Trigger before a Trigger Event.

Configuration

External register TRBTRG_EL1 bits [63:0] are architecturally mapped to AArch64 System register [TRBTRG_EL1\[63:0\]](#).

This register is present only when FEAT_TRBE_EXT is implemented. Otherwise, direct accesses to TRBTRG_EL1 are RES0.

Attributes

TRBTRG_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																RES0															
																TRG															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:32]

Reserved, RES0.

TRG, bits [31:0]

Trigger count.

Specifies the number of bytes of trace to capture following a Detected Trigger before a Trigger Event.

[TRBTRG_EL1](#) decrements by 1 for every byte of trace written to the trace buffer when all of the following are true:

- [TRBTRG_EL1](#) is nonzero.
- [TRBSR_EL1](#).TRG is 1.

The architecture places restrictions on the values that software can write to the counter.

Note

As a result of the restrictions an implementation might treat some of TRG[M:0] as RES0, where M is defined by [TRBIDR_EL1](#).Align.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing TRBTRG_EL1

The PE might ignore a write to [TRBTRG_EL1](#) if any of the following apply:

- [TRBLIMITR_EL1](#).E == 1, and either FEAT_TRBE_EXT is not implemented or the Trace Buffer Unit is using Self-hosted mode.
- [TRBLIMITR_EL1](#).XE == 1, FEAT_TRBE_EXT is implemented, and the Trace Buffer Unit is using External mode.

TRBTRG_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
TRBE	0x020	TRBTRG_EL1

Accesses to this interface are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL1, bit [17]**When TRCIDR6.EXLEVEL_RL_EL1 == 1:**

Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state.

EXLEVEL_RL_EL1	Meaning
0b0	When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1.
0b1	When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL0, bit [16]**When TRCIDR6.EXLEVEL_RL_EL0 == 1:**

Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state.

EXLEVEL_RL_EL0	Meaning
0b0	When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0.
0b1	When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [15]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [14]**When Non-secure EL2 is implemented:**

Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state.

EXLEVEL_NS_EL2	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL2.
0b1	The Address Comparator does not perform comparisons in Non-secure EL2.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL1, bit [13]**When Non-secure EL1 is implemented:**

Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state.

EXLEVEL_NS_EL1	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL1.
0b1	The Address Comparator does not perform comparisons in Non-secure EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL0, bit [12]**When Non-secure EL0 is implemented:**

Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state.

EXLEVEL_NS_EL0	Meaning
0b0	The Address Comparator performs comparisons in Non-secure EL0.
0b1	The Address Comparator does not perform comparisons in Non-secure EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL3, bit [11]**When EL3 is implemented:**

EL3 address comparison control. Controls whether a comparison can occur at EL3.

EXLEVEL_S_EL3	Meaning
0b0	The Address Comparator performs comparisons in EL3.
0b1	The Address Comparator does not perform comparisons in EL3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL2, bit [10]**When EL2 is implemented and FEAT_SEL2 is implemented:**

Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state.

EXLEVEL_S_EL2	Meaning
0b0	The Address Comparator performs comparisons in Secure EL2.
0b1	The Address Comparator does not perform comparisons in Secure EL2.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL1, bit [9]**When Secure EL1 is implemented:**

Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state.

EXLEVEL_S_EL1	Meaning
0b0	The Address Comparator performs comparisons in Secure EL1.
0b1	The Address Comparator does not perform comparisons in Secure EL1.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_ELO, bit [8]**When Secure EL0 is implemented:**

Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state.

EXLEVEL_S_ELO	Meaning
0b0	The Address Comparator performs comparisons in Secure EL0.
0b1	The Address Comparator does not perform comparisons in Secure EL0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [7]

Reserved, RES0.

CONTEXT, bits [6:4]**When TRCIDR4.NUMCIDC != 0b0000 or TRCIDR4.NUMVMIDC != 0b0000:**

Selects a Context Identifier Comparator or Virtual Context Identifier Comparator:

CONTEXT	Meaning	Applies when
0b000	Comparator 0.	
0b001	Comparator 1.	When UInt(TRCIDR4.NUMCIDC) > 1 or UInt(TRCIDR4.NUMVMIDC) > 1
0b010	Comparator 2.	When UInt(TRCIDR4.NUMCIDC) > 2 or UInt(TRCIDR4.NUMVMIDC) > 2
0b011	Comparator 3.	When UInt(TRCIDR4.NUMCIDC) > 3 or UInt(TRCIDR4.NUMVMIDC) > 3
0b100	Comparator 4.	When UInt(TRCIDR4.NUMCIDC) > 4 or UInt(TRCIDR4.NUMVMIDC) > 4
0b101	Comparator 5.	When UInt(TRCIDR4.NUMCIDC) > 5 or UInt(TRCIDR4.NUMVMIDC) > 5
0b110	Comparator 6.	When UInt(TRCIDR4.NUMCIDC) > 6 or UInt(TRCIDR4.NUMVMIDC) > 6
0b111	Comparator 7.	When UInt(TRCIDR4.NUMCIDC) > 7 or UInt(TRCIDR4.NUMVMIDC) > 7

The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CONTEXTTYPE, bits [3:2]**When TRCIDR4.NUMCIDC != 0b0000 or TRCIDR4.NUMVMIDC != 0b0000:**

Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons.

CONTEXTTYPE	Meaning	Applies when
0b00	The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators.	
0b01	The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match.	When TRCIDR4.NUMCIDC != 0b0000
0b10	The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match.	When TRCIDR4.NUMVMIDC != 0b0000
0b11	The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match.	When TRCIDR4.NUMCIDC != 0b0000 and TRCIDR4.NUMVMIDC != 0b0000

If [TRCIDR4.NUMCIDC](#) == 0b0000, then bit [2] is RES0.

If [TRCIDR4.NUMVMIDC](#) == 0b0000, then bit [3] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCACVR<n>, Address Comparator Value Register <n>, n = 0 - 15

The TRCACVR<n> characteristics are:

Purpose

Contains the address value.

Configuration

External register TRCACVR<n> bits [63:0] are architecturally mapped to AArch64 System register [TRCACVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $\text{UInt}(\text{TRCIDR4.NUMACPAIRS}) * 2 > n$. Otherwise, direct accesses to TRCACVR<n> are RES0.

Attributes

TRCACVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ADDRESS															
																ADDRESS															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADDRESS, bits [63:0]

Address Value.

The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>.

The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an UNKNOWN value, where P is defined as the maximum virtual address size supported by the PE.

The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCACVR<n>

Must be programmed if any of the following are true:

- [TRCBBCTLR](#).RANGE[n/2] == 1.
- [TRCRSCTLR<a>](#).GROUP == 0b0100 and [TRCRSCTLR<a>](#).SAC[n] == 1.
- [TRCRSCTLR<a>](#).GROUP == 0b0101 and [TRCRSCTLR<a>](#).ARC[n/2] == 1.

- [TRCVIIECTLR](#).EXCLUDE[n/2] == 1.
- [TRCVIIECTLR](#).INCLUDE[n/2] == 1.
- [TRCVISSCTLR](#).START[n] == 1.
- [TRCVISSCTLR](#).STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- [TRCQCTLR](#).RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCACVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x400 + (8 * n)	TRCACVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Configuration

External register TRCAUTHSTATUS bits [31:0] are architecturally mapped to AArch64 System register [TRCAUTHSTATUS\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCAUTHSTATUS are RES0.

Attributes

TRCAUTHSTATUS is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				RTNID	RTID	RES0				RLNID	RLID	HNID	HID	SNID	SID	NSNID	NSID														

Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

RTID	Meaning
0b00	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RLNID.

RLID, bits [13:12]

Realm invasive debug.

RLID	Meaning
0b00	Not implemented.

HNID, bits [11:10]

Hyp Non-invasive Debug. Indicates whether a separate enable control for EL2 non-invasive debug features is implemented and enabled.

HNID	Meaning
0b00	Separate Hyp non-invasive debug enable not implemented, or EL2 non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

HID, bits [9:8]

Hyp Invasive Debug. Indicates whether a separate enable control for EL2 invasive debug features is implemented and enabled.

HID	Meaning
0b00	Separate Hyp invasive debug enable not implemented, or EL2 invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

SNID, bits [7:6]

Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled.

SNID	Meaning
0b00	Secure non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b00.

When EL3 is not implemented and the PE is Secure, this field takes the value 0b10 or 0b11 depending whether Secure non-invasive debug is enabled.

SID, bits [5:4]

Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled.

SID	Meaning
0b00	Secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

NSNID, bits [3:2]

Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled.

NSNID	Meaning
0b00	Non-secure non-invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

When EL3 is implemented, this field reads as 0b11.

When EL3 is not implemented and the PE is Non-secure, this field reads as 0b11.

When EL3 is not implemented and the PE is Secure, this field reads as 0b00.

NSID, bits [1:0]

Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled.

NSID	Meaning
0b00	Non-secure invasive debug features not implemented.
0b10	Implemented and disabled.
0b11	Implemented and enabled.

All other values are reserved.

This field reads as 0b00.

Accessing TRCAUTHSTATUS

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

External debugger accesses to this register are unaffected by the OS Lock.

TRCAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFB8	TRCAUTHSTATUS

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCAUXCTLR, Auxiliary Control Register

The TRCAUXCTLR characteristics are:

Purpose

The function of this register is IMPLEMENTATION DEFINED.

Configuration

External register TRCAUXCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCAUXCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCAUXCTLR are RES0.

Attributes

TRCAUXCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPLEMENTATION DEFINED																															

IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

This field reads as an IMPLEMENTATION DEFINED value and writes to this field have IMPLEMENTATION DEFINED behavior.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Accessing TRCAUXCTLR

If this register is nonzero then it might cause the behavior of a trace unit to contradict this architecture specification. See the documentation of the specific implementation for information about the IMPLEMENTATION DEFINED support for this register.

TRCAUXCTLR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x018	TRCAUXCTLR

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCBBCTLR, Branch Broadcast Control Register

The TRCBBCTLR characteristics are:

Purpose

Controls the regions in the memory map where branch broadcasting is active.

Configuration

External register TRCBBCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCBBCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented**, TRCIDR0.TRCBB == 1 and UInt(TRCIDR4.NUMACPAIRS) > 0. Otherwise, direct accesses to TRCBBCTLR are RES0.

Attributes

TRCBBCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										MODE	RANGE[7]	RANGE[6]	RANGE[5]	RANGE[4]	RANGE[3]	RANGE[2]	RANGE[1]	RANGE[0]													

Bits [31:9]

Reserved, RES0.

MODE, bit [8]

Mode.

MODE	Meaning
0b0	Exclude Mode. Branch broadcasting is not active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then branch broadcasting is active for all instructions.
0b1	Include Mode. Branch broadcasting is active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then the behavior of the trace unit is CONSTRAINED UNPREDICTABLE. That is, the trace unit might or might not consider any instructions to be in a branch broadcasting region.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RANGE[<m>], bit [m], for m = 7 to 0

Address range field.

Selects whether Address Range Comparator <m> is used with branch broadcasting.

RANGE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected.
0b1	The address range that Address Range Comparator <m> defines, is selected.

This bit is RES0 if m >= [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCBBCTLR

Must be programmed if [TRCCONFIGR](#).BB == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCBBCTLR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x03C	TRCBBCTLR

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCCCTLR characteristics are:

Purpose

Set the threshold value for cycle counting.

Configuration

External register TRCCCTL bits [31:0] are architecturally mapped to AArch64 System register [TRCCCTL\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR0.TRCCCI == 1. Otherwise, direct accesses to TRCCCCTLR are RES0.

Attributes

TRCCCCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
											RESO																	THRESHOLD																

Bits [31:12]

Reserved, RES0.

THRESHOLD, bits [11:0]

Sets the threshold value for instruction trace cycle counting.

The minimum threshold value that can be programmed into THRESHOLD is given in [TRCIDR3.CCITMIN](#). If the THRESHOLD value is smaller than the value in [TRCIDR3.CCITMIN](#) then the behavior is CONSTRAINED UNPREDICTABLE. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known.

Writing a value of zero when [TRCCONFIGR](#).CCI enables instruction trace cycle counting results in CONSTRAINED UNPREDICTABLE behavior. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCCTLR

Must be programmed if [TRCCONFIGR.CCI](#) == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCCCTLR can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

TRCCCCTLR, Cycle Count Control Register

ETE	0x038	TRCCCCTLR
-----	-------	-----------

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCCTLR0, Context Identifier Comparator Control Register 0

The TRCCIDCCTLR0 characteristics are:

Purpose

Contains Context identifier mask values for the [TRCCIDCVR<n>](#) registers, for n = 0 to 3.

Configuration

External register TRCCIDCCTLR0 bits [31:0] are architecturally mapped to AArch64 System register [TRCCIDCCTLR0\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented**, $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 0x0$ and $\text{UInt}(\text{TRCIDR2.CIDSIZE}) > 0$. Otherwise, direct accesses to TRCCIDCCTLR0 are RES0.

Attributes

TRCCIDCCTLR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	
COMP3[7]	COMP3[6]	COMP3[5]	COMP3[4]	COMP3[3]	COMP3[2]	COMP3[1]	COMP3[0]	COMP2[7]	COMP2[6]	COMP2[5]	COMP2[4]

COMP3[<m>], bit [m+24], for m = 7 to 0
When $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 3$:

TRCCIDCVR3 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR3. Each bit in this field corresponds to a byte in TRCCIDCVR3.

COMP3[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR3[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR3[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP2[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 2:

TRCCIDCVR2 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR2. Each bit in this field corresponds to a byte in TRCCIDCVR2.

COMP2[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR2[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR2[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP1[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 1:

TRCCIDCVR1 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR1. Each bit in this field corresponds to a byte in TRCCIDCVR1.

COMP1[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR1[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR1[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP0[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 0:

TRCCIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0.

COMP0[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.CIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCCIDCCTRL0

If software uses the [TRCCIDCVR<n>](#) registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCCIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in **TRCCIDCVR<n>** is not 0x00, the behavior of the Context Identifier Comparator is **CONSTRAINED UNPREDICTABLE**. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCCIDCTL0 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x680	TRCCIDCCTLR0

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCCTLR1, Context Identifier Comparator Control Register 1

The TRCCIDCCTLR1 characteristics are:

Purpose

Contains Context identifier mask values for the [TRCCIDCVR<n>](#) registers, for n = 4 to 7.

Configuration

External register TRCCIDCCTLR1 bits [31:0] are architecturally mapped to AArch64 System register [TRCCIDCCTLR1\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented**, $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 0x4$ and $\text{UInt}(\text{TRCIDR2.CIDSIZE}) > 0$. Otherwise, direct accesses to TRCCIDCCTLR1 are RES0.

Attributes

TRCCIDCCTLR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	
COMP7[7]	COMP7[6]	COMP7[5]	COMP7[4]	COMP7[3]	COMP7[2]	COMP7[1]	COMP7[0]	COMP6[7]	COMP6[6]	COMP6[5]	COMP6[4]

COMP7[<m>], bit [m+24], for m = 7 to 0
When $\text{UInt}(\text{TRCIDR4.NUMCIDC}) > 7$:

TRCCIDCVR7 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR7. Each bit in this field corresponds to a byte in TRCCIDCVR7.

COMP7[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR7[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR7[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP6[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 6:

TRCCIDCVR6 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR6. Each bit in this field corresponds to a byte in TRCCIDCVR6.

COMP6[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR6[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR6[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP5[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 5:

TRCCIDCVR5 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR5. Each bit in this field corresponds to a byte in TRCCIDCVR5.

COMP5[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR5[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR5[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.CIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP4[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMCIDC) > 4:

TRCCIDCVR4 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR4. Each bit in this field corresponds to a byte in TRCCIDCVR4.

COMP4[<m>]	Meaning
0b0	The trace unit includes TRCCIDCVR4[(m×8+7):(m×8)] when it performs the Context identifier comparison.
0b1	The trace unit ignores TRCCIDCVR4[(m×8+7):(m×8)] when it performs the Context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.CIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCCIDCTRL1

If software uses the [TRCCIDCVR<n>](#) registers, for n = 4 to 7, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCCIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in **TRCCIDCVR<n>** is not 0x00, the behavior of the Context Identifier Comparator is **CONSTRAINED UNPREDICTABLE**. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCCIDCTLR1 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x684	TRCCIDCCTLR1

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n>, n = 0 - 7

The TRCCIDCVR<n> characteristics are:

Purpose

Contains a Context identifier value.

Configuration

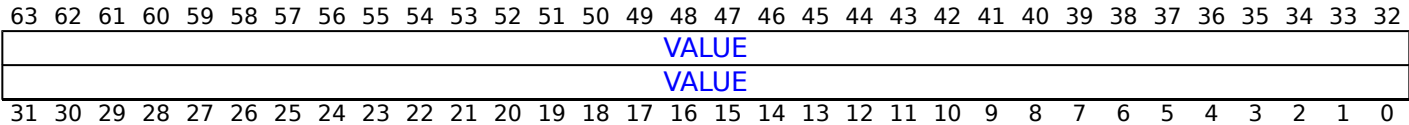
External register TRCCIDCVR<n> bits [63:0] are architecturally mapped to AArch64 System register [TRCCIDCVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR4.NUMCIDC) > n$. Otherwise, direct accesses to TRCCIDCVR<n> are RES0.

Attributes

TRCCIDCVR<n> is a 64-bit register.

Field descriptions



VALUE, bits [63:0]

Context identifier value. The width of this field is indicated by [TRCIDR2.CIDSIZE](#). Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCIDCVR<n>

Must be programmed if any of the following are true:

- [TRCRSCTLR<a>](#).GROUP == 0b0110 and [TRCRSCTLR<a>](#).CID[n] == 1.
- [TRCACATR<a>](#).CONTEXTTYPE == 0b01 or 0b11 and [TRCACATR<a>](#).CONTEXT == n.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCCIDCVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x600 + (8 * n)	TRCCIDCVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCCIDR0, Component Identification Register 0

The TRCCIDR0 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCCIDR0 are RES0.

Attributes

TRCCIDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_0							

Bits [31:8]

Reserved, RES0.

PRMBL_0, bits [7:0]

Component identification preamble, segment 0.

Reads as 0x0D.

Access to this field is **RO**.

Accessing TRCCIDR0

External debugger accesses to this register are unaffected by the OS Lock.

TRCCIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFF0	TRCCIDR0

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

This interface is accessible as follows:

- Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

TRCCIDR2, Component Identification Register 2

The TRCCIDR2 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCCIDR2 are RES0.

Attributes

TRCCIDR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_2							

Bits [31:8]

Reserved, RES0.

PRMBL_2, bits [7:0]

Component identification preamble, segment 2.

Reads as 0x05.

Access to this field is **RO**.

Accessing TRCCIDR2

External debugger accesses to this register are unaffected by the OS Lock.

TRCCIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFF8	TRCCIDR2

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCCIDR3, Component Identification Register 3

The TRCCIDR3 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCCIDR3 are RES0.

Attributes

TRCCIDR3 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_3							

Bits [31:8]

Reserved, RES0.

PRMBL_3, bits [7:0]

Component identification preamble, segment 3.

Reads as 0xB1.

Access to this field is **RO**.

Accessing TRCCIDR3

External debugger accesses to this register are unaffected by the OS Lock.

TRCCIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFFC	TRCCIDR3

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCCLAIMSET, Claim Tag Set Register

The TRCCLAIMSET characteristics are:

Purpose

In conjunction with [TRCCLAIMCLR](#), provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configuration

External register TRCCLAIMSET bits [31:0] are architecturally mapped to AArch64 System register [TRCCLAIMSET\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCCLAIMSET are RES0.

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

Attributes

TRCCLAIMSET is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18		
SET[31]	SET[30]	SET[29]	SET[28]	SET[27]	SET[26]	SET[25]	SET[24]	SET[23]	SET[22]	SET[21]	SET[20]	SET[19]	SET[18]	SE	

SET[<m>], bit [m], for m = 31 to 0

Claim Tag Set. Indicates whether Claim Tag bit <m> is implemented, and is used to set Claim Tag bit <m> to 1.

SET[<m>]	Meaning
0b0	On a read: Claim Tag bit <m> is not implemented. On a write: Ignored.
0b1	On a read: Claim Tag bit <m> is implemented. On a write: Set Claim Tag bit <m> to 1.

This bit reads-as-zero and ignores writes if m > the number of Claim Tag bits.

Access to this field is **RAO/W1S**.

Accessing TRCCLAIMSET

TRCCLAIMSET can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFA0	TRCCLAIMSET

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.

- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

RLDSELF	Meaning
0b0	Normal mode. The Counter is in Normal mode.
0b1	Self-reload mode. The Counter is in Self-reload mode.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RLDEVENT_TYPE, bit [15]

Chooses the type of Resource Selector.

Selects an event, that when it occurs causes a reload event for Counter <n>.

RLDEVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [14:13]

Reserved, RES0.

RLDEVENT_SEL, bits [12:8]

Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Selects an event, that when it occurs causes a reload event for Counter <n>.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

CNTEVENT_TYPE, bit [7]

Chooses the type of Resource Selector.

Selects an event, that when it occurs causes Counter <n> to decrement.

CNTEVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

CNTEVENT_SEL, bits [4:0]

Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Selects an event, that when it occurs causes Counter <n> to decrement.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCNTCTL<n>

Must be programmed if `TRCRSCTLR<a>.GROUP == 0b0010` and `TRCRSCTLR<a>.COUNTERS[n] == 1`.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCNTCTLR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x150 + (4 * n)	TRCCNTCTLR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

(old)

htmldiff from-

(new)

TRCCNTRLDVR<n>, Counter Reload Value Register <n>, n = 0 - 3

The TRCCNTRLDVR<n> characteristics are:

Purpose

This sets or returns the reload count value for Counter <n>.

Configuration

External register TRCCNTRLDVR<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCCNTRLDVR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR5.NUMCNTR) > n$. Otherwise, direct accesses to TRCCNTRLDVR<n> are RES0.

Attributes

TRCCNTRLDVR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																VALUE															

Bits [31:16]

Reserved, RES0.

VALUE, bits [15:0]

Contains the reload value for Counter <n>. When a reload event occurs for Counter <n> then the trace unit copies the VALUE<n> field into Counter <n>.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCCNTRLDVR<n>

Must be programmed if [TRCRSCTLR<a>](#).GROUP == 0b0010 and [TRCRSCTLR<a>](#).COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCCNTRLDVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	$0 \times 140 + (4 * n)$	TRCCNTRLDVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCCNTVR<n> characteristics are:

This sets or returns the value of Counter <n>.

External register TRCCNTVR<n> bits [31:0] are architecturally mapped to AArch64 System register TRCCNTVR<n>[31:0].

This register is present only when FEAT_ETE is implemented, FEAT_TRC_EXT is implemented and UInt(TRCIDR5.NUMCNTR) > n. Otherwise, direct accesses to TRCCNTVR<n> are RES0.

TRCCNTVR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																VALUE															

Bits [31:16]

Reserved, RES0.

VALUE, bits [15:0]

Contains the count value of Counter.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCNTVR<n>

Must be programmed if [TRCRSCTLR<a>](#).GROUP == 0b0010 and [TRCRSCTLR<a>](#).COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

TRCCNTVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	$0 \times 160 + (4 * n)$	TRCCNTVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.

- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCCONFIGR, Trace Configuration Register

The TRCCONFIGR characteristics are:

Purpose

Controls the tracing options.

Configuration

External register TRCCONFIGR bits [31:0] are architecturally mapped to AArch64 System register [TRCCONFIGR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCCONFIGR are RES0.

Attributes

TRCCONFIGR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19		18		17		16		15		14	13		12		11		10	9	8		7		6		5		

Bits [31:19:16]

Reserved, RES0.

ITO, bit [18]

When TRCIDR0.ITE == 1:

Instrumentation Trace Override.

ITO	Meaning
0b0	Instrumentation Trace Override disabled.
0b1	Instrumentation Trace Override enabled.

This field is ignored when SelfHostedTraceEnabled() returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [17:16]

Reserved, RES0.

VMIDOPT, bit [15]**When TRCIDR2.VMIDOPT == 0b01:**

Virtual context identifier selection control.

VMIDOPT	Meaning
0b0	VTTBR_EL2 .VMID is used as the Virtual context identifier.
0b1	CONTEXTIDR_EL2 .PROCID is used as the Virtual context identifier.

When TRCIDR2.VMIDOPT == 0b00:

Reserved, RES0.

Virtual context identifier selection control.

[VTTBR_EL2](#).VMID is used as the Virtual context identifier.

When TRCIDR2.VMIDOPT == 0b10:

Reserved, RES1.

Virtual context identifier selection control.

[CONTEXTIDR_EL2](#).PROCID is used as the Virtual context identifier.

Otherwise:

Reserved, RES0.

QE, bits [14:13]**When TRCIDR0.QSUPP == 0b01:**

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b01	Q elements with instruction counts are enabled.
	Q elements without instruction counts are disabled.

All other values are reserved.

When TRCIDR0.QSUPP == 0b10:

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b11	Q elements with instruction counts are enabled.
	Q elements without instruction counts are enabled.

All other values are reserved.

When TRCIDR0.QSUPP == 0b11:

Q element generation control.

QE	Meaning
0b00	Q elements are disabled.
0b01	Q elements with instruction counts are enabled. Q elements without instruction counts are disabled.
0b11	Q elements with instruction counts are enabled. Q elements without instruction counts are enabled.

All other values are reserved.

Otherwise:

Reserved, RES0.

RS, bit [12]**When TRCIDR0.RETSTACK == 1:**

Return stack control.

RS	Meaning
0b0	Return stack is disabled.
0b1	Return stack is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TS, bit [11]**When TRCIDR0.TSSIZE != 0b00000:**

Global timestamp tracing control.

TS	Meaning
0b0	Global timestamp tracing is disabled.
0b1	Global timestamp tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [10:8]

Reserved, RES0.

VMID, bit [7]**When TRCIDR2.VMIDSIZE != 0b000000:**

Virtual context identifier tracing control.

VMID	Meaning
0b0	Virtual context identifier tracing is disabled.
0b1	Virtual context identifier tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

CID, bit [6]**When TRCIDR2.CIDSIZE != 0b000000:**

Context identifier tracing control.

CID	Meaning
0b0	Context identifier tracing is disabled.
0b1	Context identifier tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [5]

Reserved, RES0.

CCI, bit [4]**When TRCIDR0.TRCCCI == 1:**

Cycle counting instruction tracing control.

CCI	Meaning
0b0	Cycle counting instruction tracing is disabled.
0b1	Cycle counting instruction tracing is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

The reset behavior of this field is:

Reserved, RES1.

(old)

htmldiff from-

(new)

TRCDEVAFF, Device Affinity Register

The TRCDEVAFF characteristics are:

Purpose

For additional information, see the CoreSight Architecture Specification.

Reads the same value as the [MPIDR_EL1](#) register for the PE that this trace unit has affinity with.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCDEVAFF are RES0.

Attributes

TRCDEVAFF is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																MPIDR_EL1															
																MPIDR_EL1															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MPIDR_EL1, bits [63:0]

Read-only copy of [MPIDR_EL1](#), as seen from the highest implemented Exception level.

Accessing TRCDEVAFF

External debugger accesses to this register are unaffected by the OS Lock.

TRCDEVAFF can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFA8	TRCDEVAFF

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCDEVARCH, Device Architecture Register

The TRCDEVARCH characteristics are:

Purpose

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

External register TRCDEVARCH bits [31:0] are architecturally mapped to AArch64 System register [TRCDEVARCH\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCDEVARCH are RES0.

Attributes

TRCDEVARCH is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											

ARCHITECT, bits [31:21]

Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.

ARCHITECT	Meaning
0b01000111011	JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.

Other values are defined by the JEDEC JEP106 standard.

This field reads as 0x23B.

PRESENT, bit [20]

DEVARCH Present. Defines that the DEVARCH register is present.

PRESENT	Meaning
0b0	Device Architecture information not present.
0b1	Device Architecture information present.

This field reads as 1.

REVISION, bits [19:16]

Revision. Defines the architecture revision of the component.

REVISION	Meaning
0b0000	ETEv1.0, FEAT ETE.
0b0001	ETEv1.1, FEAT ETEv1p1.
0b0010	ETEv1.2, FEAT ETEv1p2.
0b0011	ETEv1.3, FEAT ETEv1p3.

All other values are reserved.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component.

ARCHVER	Meaning
0b0101	ETEv1.

ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12].

This field reads as 0x5.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

ARCHPART	Meaning
0xA13	Arm PE trace architecture.

ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0].

This field reads as 0xA13.

Accessing TRCDEVARCH

External debugger accesses to this register are unaffected by the OS Lock.

TRCDEVARCH can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFBC	TRCDEVARCH

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCDEVID characteristics are:

Purpose

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

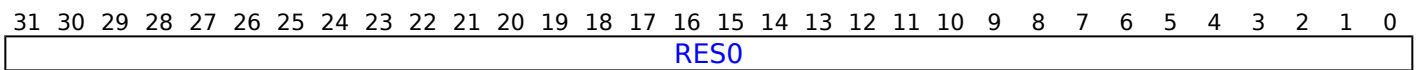
External register TRCDEVID bits [31:0] are architecturally mapped to AArch64 System register [TRCDEVID\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCDEVID are RES0.

Attributes

TRCDEVID is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCDEVID

External debugger accesses to this register are unaffected by the OS Lock.

TRCDEVID can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFC8	TRCDEVID

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)htmldiff from-(new)

TRCDEVID1, Device Configuration Register 1

The TRCDEVID1 characteristics are:

Purpose

Provides discovery information for the component.
For additional information, see the CoreSight Architecture Specification.

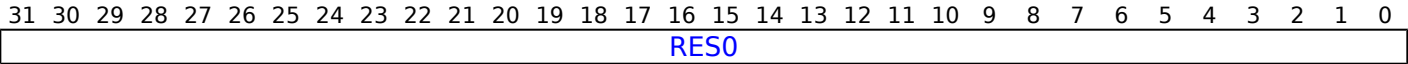
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCDEVID1 are RES0.

Attributes

TRCDEVID1 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCDEVID1

External debugger accesses to this register are unaffected by the OS Lock.

TRCDEVID1 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFC4	TRCDEVID1

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)htmldiff from-(new)

TRCDEVID2, Device Configuration Register 2

The TRCDEVID2 characteristics are:

Purpose

Provides discovery information for the component.
For additional information, see the CoreSight Architecture Specification.

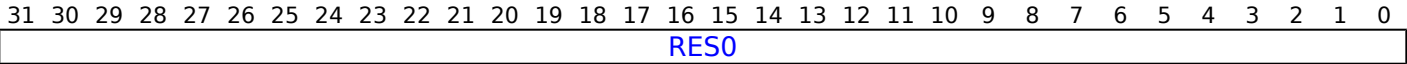
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCDEVID2 are RES0.

Attributes

TRCDEVID2 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCDEVID2

External debugger accesses to this register are unaffected by the OS Lock.

TRCDEVID2 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFC0	TRCDEVID2

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

(old)

htmldiff from-

(new)

TRCDEVTYPE, Device Type Register

The TRCDEVTYPE characteristics are:

Purpose

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCDEVTYPE are RES0.

Attributes

TRCDEVTYPE is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0												SUB			MAJOR				

Bits [31:8]

Reserved, RES0.

SUB, bits [7:4]

Component sub-type.

SUB	Meaning
0b0001	When MAJOR == 0x3 (Trace source): Associated with a PE.

This field reads as 0x1.

MAJOR, bits [3:0]

Component major type.

MAJOR	Meaning
0b0011	Trace source.

Other values are defined by the CoreSight Architecture.

This field reads as 0x3.

Accessing TRCDEVTYPE

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance
ETE	0xFCC	TRCDEVTYPE

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCEVENTCTL0R, Event Control 0 Register

The TRCEVENTCTL0R characteristics are:

Purpose

Controls the generation of ETEEvents.

Configuration

External register TRCEVENTCTL0R bits [31:0] are architecturally mapped to AArch64 System register [TRCEVENTCTL0R\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR4.NUMRSPAIR != 0b0000. Otherwise, direct accesses to TRCEVENTCTL0R are RES0.

Attributes

TRCEVENTCTL0R is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
EVENT3_TYPE	RES0	EVENT3_SEL	EVENT2_TYPE	RES0	EVENT2_SEL	EVENT1_TYPE	RES0	EVENT1_SEL	EVENT0_TYPE	RES0	EVENT0_SEL	EVENT0_TYPE	RES0	EVENT0_SEL	EVENT0_TYPE	RES0	EVENT0_SEL	EVENT0_TYPE	RES0	EVENT0_SEL	EVENT0_TYPE	RES0	EVENT0_SEL	EVENT0_TYPE	RES0	EVENT0_SEL

EVENT3_TYPE, bit [31]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 3:

Chooses the type of Resource Selector.

EVENT3_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT3.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT3.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [30:29]

Reserved, RES0.

EVENT3_SEL, bits [28:24]**When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 3:**

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.TYPE controls whether TRCEVENTCTL0R.EVENT3.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R.INSTEN\[3\]](#) == 1, then Event element 3 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT2_TYPE, bit [23]**When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 2:**

Chooses the type of Resource Selector.

EVENT2_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT2.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT2.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [22:21]

Reserved, RES0.

EVENT2_SEL, bits [20:16]**When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 2:**

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.TYPE controls whether TRCEVENTCTL0R.EVENT2.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R](#).INSTEN[2] == 1, then Event element 2 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT1_TYPE, bit [15]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 1:

Chooses the type of Resource Selector.

EVENT1_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT1.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT1.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [14:13]

Reserved, RES0.

EVENT1_SEL, bits [12:8]

When TRCIDR4.NUMRSPAIR != 0b0000 and UInt(TRCIDR0.NUMEVENT) >= 1:

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.TYPE controls whether TRCEVENTCTL0R.EVENT1.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R](#).INSTEN[1] == 1, then Event element 1 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EVENT0_TYPE, bit [7]

When TRCIDR4.NUMRSPAIR != 0b0000:

Chooses the type of Resource Selector.

EVENT0_TYPE	Meaning
0b0	A single Resource Selector. TRCEVENTCTL0R.EVENT0.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT0.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [6:5]

Reserved, RES0.

EVENT0_SEL, bits [4:0]

When TRCIDR4.NUMRSPAIR != 0b0000:

Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.TYPE controls whether TRCEVENTCTL0R.EVENT0.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

When any of the selected resource events occurs and [TRCEVENTCTL1R.INSTEN\[0\]](#) == 1, then Event element 0 is generated in the instruction trace element stream.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance
ETE	0x020	TRCEVENTCTL0R

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

Page 3343

(old)

htmldiff from-

(new)

TRCEVENTCTL1R, Event Control 1 Register

The TRCEVENTCTL1R characteristics are:

Purpose

Controls the behavior of the ETEEvents that [TRCEVENTCTL0R](#) selects.

Configuration

External register TRCEVENTCTL1R bits [31:0] are architecturally mapped to AArch64 System register [TRCEVENTCTL1R\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCEVENTCTL1R are RES0.

Attributes

TRCEVENTCTL1R is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2		
RES0														OE		LPOVERRIDE		LPOVERRIDE		ATB		ATB		RES0		INSTEN[3]		INSTEN[32]		INSTEN[31]	

Bits [31:14]13

Reserved, RES0.

OE, bit [13]

When TRCIDR5.OE == 1:

ETE Trace Output Enable control.

OE	Meaning
0b0	Trace output to any IMPLEMENTATION DEFINED trace output interface is disabled.
0b1	Trace output to any IMPLEMENTATION DEFINED trace output interface is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Otherwise:

Reserved, RES0.

LPOVERRIDE, bit [12]

When TRCIDR5.LPOVERRIDE == 1:

Low-power Override Mode select.

LPOVERRIDE	Meaning
0b0	Trace unit Low-power Override Mode is not enabled. That is, the trace unit is permitted to enter low-power state.
0b1	Trace unit Low-power Override Mode is enabled. That is, entry to a low-power state does not affect the trace unit resources or trace generation.

Otherwise:

Reserved, RES0.

ATB, bit [11]**When TRCIDR5.ATBTRIG == 1:**

AMBA Trace Bus (ATB) trigger enable.

If a CoreSight ATB interface is implemented then when ETEEvent 0 occurs the trace unit sets:

- ATID == 0x7D.
- ATDATA to the value of [TRCTRACEIDR](#).

If the width of ATDATA is greater than the width of [TRCTRACEIDR](#).TRACEID then the trace unit zeros the upper ATDATA bits.

If ETEEvent 0 is programmed to occur based on program execution, such as an Address Comparator, the ATB trigger might not be inserted into the ATB stream at the same time as any trace generated by that program execution is output by the trace unit. Typically, the generated trace might be buffered in a trace unit which means that the ATB trigger would be output before the associated trace is output.

If ETEEvent 0 is asserted multiple times in close succession, the trace unit is required to generate an ATB trigger for the first assertion, but might ignore one or more of the subsequent assertions. Arm recommends that the window in which ETEEvent 0 is ignored is limited only by the time taken to output an ATB trigger.

ATB	Meaning
0b0	ATB trigger is disabled.
0b1	ATB trigger is enabled.

Otherwise:

Reserved, RES0.

Bits [10:4]

Reserved, RES0.

INSTEN[<m>], bit [m], for m = 3 to 0

Event element control.

INSTEN[<m>]	Meaning
0b0	The trace unit does not generate an Event element <m>.
0b1	The trace unit generates an Event element <m>.

This bit is RES0 if m >= the number indicated by [TRCIDR0](#).NUMEVENT.

Accessing TRCEVENTCTL1R

Must be programmed.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCEVENTCTL1R can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x024	TRCEVENTCTL1R

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCEXTINSELN<n>, External Input Select Register <n>, n = 0 - 3

The TRCEXTINSELN<n> characteristics are:

Purpose

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELN is an alias of TRCEXTINSELN0.

Configuration

External register TRCEXTINSELN<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCEXTINSELN<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR5.NUMEXTINSEL) > n$. Otherwise, direct accesses to TRCEXTINSELN<n> are RES0.

Attributes

TRCEXTINSELN<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																evtCount															

Bits [31:16]

Reserved, RES0.

evtCount, bits [15:0]

PMU event to select.

The event number as defined by the Arm ARM.

Software must program this field with a PMU event that is supported by the PE being programmed.

There are three ranges of PMU event numbers:

- PMU event numbers in the range 0x0000 to 0x003F are common architectural and microarchitectural events.
- PMU event numbers in the range 0x0040 to 0x00BF are Arm recommended common architectural and microarchitectural PMU events.
- PMU event numbers in the range 0x00C0 to 0x03FF are IMPLEMENTATION DEFINED PMU events.

If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type:

- For the range 0x0000 to 0x003F, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field.
- For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

UNPREDICTABLE means the PMU event must not expose privileged information.

Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common IMPLEMENTATION DEFINED PMU events, then no PMU event is counted and the value read back on evtCount is the value written.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCEXTINSEL_R<n>

Must be programmed if any of the following is true: `TRCRSCTLR<a>.GROUP == 0b0000` and `TRCRSCTLR<a>.EXTIN[n] == 1`.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCEXTINSEL_R<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x120 + (4 * n)	TRCEXTINSEL<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR0, ID Register 0

The TRCIDR0 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR0 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR0\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR0 are RES0.

Attributes

TRCIDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
RES0	COMMTRANS	COMMOPT	TSSIZE	TSMARK	ITERES0	RES0	RES0	TRCEXDATA	TRCEXDATA	QSUPP	QSUPP	QFILT	QFILT	COND	COND	COND	COND

Bit [31]

Reserved, RES0.

COMMTRANS, bit [30]

Transaction Start element behavior.

COMMTRANS	Meaning
0b0	Transaction Start elements are P0 elements.
0b1	Transaction Start elements are not P0 elements.

COMMOPT, bit [29]

Indicates the contents and encodings of Cycle count packets.

COMMOPT	Meaning
0b0	Commit mode 0.
0b1	Commit mode 1.

The Commit mode defines the contents and encodings of Cycle Count packets, in particular how Commit elements are indicated by these packets. See the descriptions of these packets for more details.

Accessing this field has the following behavior:

- Access is **RAO/WI** if all of the following are true:
 - TRCIDR0.TRCCCI == 1
 - UInt(TRCIDR8.MAXSPEC) == 0x0
- When TRCIDR0.TRCCCI == 0, access to this field is **RAZ/WI**.
- Otherwise, access to this field is **RO**.

TSSIZE, bits [28:24]

Indicates that the trace unit implements Global timestamping and the size of the timestamp value.

TSSIZE	Meaning
0b00000	Global timestamping not implemented.
0b01000	Global timestamping implemented with a 64-bit timestamp value.

All other values are reserved.

This field reads as 0b01000.

TSMARK, bit [23]**When FEAT_ETEv1p1 is implemented:**

Indicates whether Timestamp Marker elements are generated.

TSMARK	Meaning
0b0	Timestamp Marker elements are not generated.
0b1	Timestamp Marker elements are generated.

Otherwise:

Reserved, RES0.

ITE, Bits bit [22:18]**When FEAT_ETEv1p3 is implemented:**

Indicates whether Instrumentation Trace is implemented.

ITE	Meaning
0b0	Instrumentation Trace not implemented.
0b1	Instrumentation Trace implemented.

This field has the value 1 if FEAT_ITE is implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [21:18]

Reserved, RES0.

TRCEXDATA, bit [17]**When TRCIDR0.TRCDATA != 0b00:**

Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

TRCEXDATA	Meaning
0b0	Tracing of data transfers for exceptions and exception returns not implemented.
0b1	Tracing of data transfers for exceptions and exception returns implemented.

Otherwise:

Reserved, RES0.

QSUPP, bits [16:15]

Indicates that the trace unit implements Q element support.

QSUPP	Meaning
0b00	Q element support is not implemented.
0b01	Q element support is implemented, and only supports Q elements with instruction counts.
0b10	Q element support is implemented, and only supports Q elements without instruction counts.
0b11	Q element support is implemented, and supports: <ul style="list-style-type: none"> • Q elements with instruction counts. • Q elements without instruction counts.

QFILT, bit [14]

Indicates if the trace unit implements Q element filtering.

QFILT	Meaning
0b0	Q element filtering is not implemented.
0b1	Q element filtering is implemented.

If TRCIDR0.QSUPP == 0b00 then this field is 0.

CONDTYPE, bits [13:12]**When TRCIDR0.TRCCOND == 1:**

Indicates how conditional instructions are traced. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

CONDTYPE	Meaning
0b00	Conditional instructions are traced with an indication of whether they pass or fail their condition code check.
0b01	Conditional instructions are traced with an indication of the APSR condition flags.

All other values are reserved.

Otherwise:

Reserved, RES0.

NUMEVENT, bits [11:10]**When TRCIDR4.NUMRSPAIR == 0b0000:**

Indicates the number of ETEEvents implemented.

NUMEVENT	Meaning
0b00	The trace unit supports 0 ETEEvents.

All other values are reserved.

When TRCIDR4.NUMRSPAIR != 0b0000:

Indicates the number of ETEEvents implemented.

NUMEVENT	Meaning
0b00	The trace unit supports 1 ETEEvent.
0b01	The trace unit supports 2 ETEEvents.
0b10	The trace unit supports 3 ETEEvents.
0b11	The trace unit supports 4 ETEEvents.

Otherwise:

Reserved, RES0.

RETSTACK, bit [9]

Indicates if the trace unit supports the return stack.

RETSTACK	Meaning
0b0	Return stack not implemented.
0b1	Return stack implemented.

Bit [8]

Reserved, RES0.

TRCCCI, bit [7]

Indicates if the trace unit implements cycle counting.

TRCCCI	Meaning
0b0	Cycle counting not implemented.
0b1	Cycle counting implemented.

This field reads as 1.

TRCCOND, bit [6]

Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures.

TRCCOND	Meaning
0b0	Conditional instruction tracing not implemented.
0b1	Conditional instruction tracing implemented.

This field reads as 0.

TRCBB, bit [5]

Indicates if the trace unit implements branch broadcasting.

TRCBB	Meaning
0b0	Branch broadcasting not implemented.
0b1	Branch broadcasting implemented.

This field reads as 1.

(old)

htmldiff from-

(new)

TRCIDR1, ID Register 1

The TRCIDR1 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR1 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR1\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCIDR1 are RES0.

Attributes

TRCIDR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DESIGNER								RES0								RES1				TRCARCHMAJ				TRCARCHMIN				REVISION			

DESIGNER, bits [31:24]

Indicates which company designed the trace unit. The permitted values of this field are the same as [MIDR_EL1](#).Implementer.

Bits [23:16]

Reserved, RES0.

Bits [15:12]

Reserved, RES1.

TRCARCHMAJ, bits [11:8]

Major architecture version.

TRCARCHMAJ	Meaning
0b1111	If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to TRCDEVARCH .

All other values are reserved.

This field reads as 0b1111.

TRCARCHMIN, bits [7:4]

Minor architecture version.

TRCARCHMIN	Meaning
0b1111	If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to TRCDEVARCH .

All other values are reserved.

This field reads as 0b1111.

REVISION, bits [3:0]

Implementation revision.

Returns an IMPLEMENTATION DEFINED value that identifies the revision of the trace unit.

Arm deprecates any use of this field and recommends that implementations set this field to zero.

Accessing TRCIDR1

TRCIDR1 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1E4	TRCIDR1

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCIDR10, ID Register 10

The TRCIDR10 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

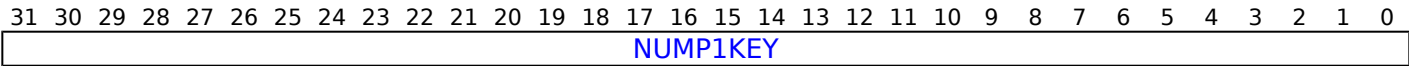
External register TRCIDR10 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR10\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCIDR10 are RES0.

Attributes

TRCIDR10 is a 32-bit register.

Field descriptions



NUMP1KEY, bits [31:0]
When TRCIDR0.TRCDATA != 0b00:

Indicates the number of P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR10

TRCIDR10 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x188	TRCIDR10

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

(old)**htmldiff from-****(new)**

(old)**htmldiff from-****(new)**

TRCIDR12, ID Register 12

The TRCIDR12 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

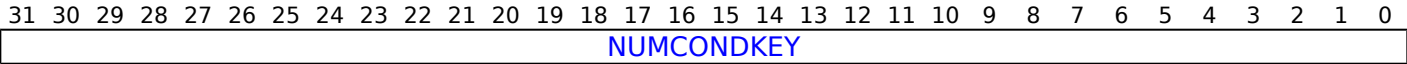
External register TRCIDR12 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR12\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCIDR12 are RES0.

Attributes

TRCIDR12 is a 32-bit register.

Field descriptions



NUMCONDKEY, bits [31:0]
When TRCIDR0.TRCCOND == 1:

Indicates the number of conditional instruction right-hand keys. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

Otherwise:

Reserved, RES0.

Accessing TRCIDR12

TRCIDR12 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x190	TRCIDR12

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

(old)**htmldiff from-****(new)**

(old)**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

TRCIDR2, ID Register 2

The TRCIDR2 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR2 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR2\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR2 are RES0.

Attributes

TRCIDR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WFXMODE	VMIDOPT	CCSIZE		DVSIZE		DASIZE		VMIDSIZE		CIDSIZE		IASIZE																			

WFXMODE, bit [31]

Indicates whether WFI, WFIT, WFE, and WFET instructions are classified as P0 instructions:

WFXMODE	Meaning
0b0	WFI, WFIT, WFE, and WFET instructions are not classified as P0 instructions.
0b1	WFI, WFIT, WFE, and WFET instructions are classified as P0 instructions.

VMIDOPT, bits [30:29]

Indicates the options for Virtual context identifier selection.

VMIDOPT	Meaning
0b00	Virtual context identifier selection not supported. TRCCONFIGR .VMIDOPT is RES0.
0b01	Virtual context identifier selection supported. TRCCONFIGR .VMIDOPT is implemented.
0b10	Virtual context identifier selection not supported. TRCCONFIGR .VMIDOPT is RES1.

All other values are reserved.

If TRCIDR2.VMIDSIZE == 0b000000 then this field is 0b00.

If TRCIDR2.VMIDSIZE != 0b000000 then this field is 0b10.

CCSIZE, bits [28:25]**When TRCIDR0.TRCCCI == 1:**

Indicates the size of the cycle counter.

CCSIZE	Meaning
0b0000	The cycle counter is 12 bits in length.
0b0001	The cycle counter is 13 bits in length.
0b0010	The cycle counter is 14 bits in length.
0b0011	The cycle counter is 15 bits in length.
0b0100	The cycle counter is 16 bits in length.
0b0101	The cycle counter is 17 bits in length.
0b0110	The cycle counter is 18 bits in length.
0b0111	The cycle counter is 19 bits in length.
0b1000	The cycle counter is 20 bits in length.

All other values are reserved.

Otherwise:

Reserved, RES0.

DVSIZE, bits [24:20]**When TRCIDR0.TRCDATA != 0b00:**

Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

DVSIZE	Meaning
0b00000	Data value tracing not implemented.
0b00100	Data value tracing has a maximum of 32-bit data values.
0b01000	Data value tracing has a maximum of 64-bit data values.

All other values are reserved.

Otherwise:

Reserved, RES0.

DASIZE, bits [19:15]**When TRCIDR0.TRCDATA != 0b00:**

Indicates the data address size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.

DASIZE	Meaning
0b00000	Data address tracing not implemented.
0b00100	Data address tracing has a maximum of 32-bit data addresses.
0b01000	Data address tracing has a maximum of 64-bit data addresses.

All other values are reserved.

Otherwise:

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCIDR3, ID Register 3

The TRCIDR3 characteristics are:

Purpose

Returns the base architecture of the trace unit.

Configuration

External register TRCIDR3 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR3\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR3 are RES0.

Attributes

TRCIDR3 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20
NOOVERFLOW	NUMPROC[2:0]	SYSSTALL	STALLCTL	SYNCPR	TRCERR	RES0	EXLEVEL_NS_EL2	EXLEVEL_NS_EL1	EXLEVEL_NS_EL0	EXLEVEL_NS_EL3	EXLEVEL_NS_EL4

NOOVERFLOW, bit [31]

Indicates if overflow prevention is implemented.

NOOVERFLOW	Meaning
0b0	Overflow prevention is not implemented.
0b1	Overflow prevention is implemented.

If TRCIDR3.STALLCTL == 0 then this field is 0.

NUMPROC, bits [13:12, 30:28]

Indicates the number of PEs available for tracing.

NUMPROC	Meaning
0b00000	The trace unit can trace one PE.

This field reads as 0b00000.

The NUMPROC field is split as follows:

- NUMPROC[2:0] is TRCIDR3[30:28].
- NUMPROC[4:3] is TRCIDR3[13:12].

SYSSTALL, bit [27]

Indicates if stalling of the PE is permitted.

SYSSTALL	Meaning
0b0	Stalling of the PE is not permitted.
0b1	Stalling of the PE is permitted.

The value of this field might be dynamic and change based on system conditions.

If TRCIDR3.STALLCTL == 0 then this field is 0.

STALLCTL, bit [26]

Indicates if trace unit implements stalling of the PE.

STALLCTL	Meaning
0b0	Stalling of the PE is not implemented.
0b1	Stalling of the PE is implemented.

SYNCPR, bit [25]

Indicates if an implementation has a fixed synchronization period.

SYNCPR	Meaning
0b0	TRCSYNCPR is read/write so software can change the synchronization period.
0b1	TRCSYNCPR is read-only so the synchronization period is fixed.

This field reads as 0.

TRCERR, bit [24]

Indicates forced tracing of System Error exceptions is implemented.

TRCERR	Meaning
0b0	Forced tracing of System Error exceptions is not implemented.
0b1	Forced tracing of System Error exceptions is implemented.

This field reads as 1.

Bit [23]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [22]

Indicates if Non-secure EL2 is implemented.

EXLEVEL_NS_EL2	Meaning
0b0	Non-secure EL2 is not implemented.
0b1	Non-secure EL2 is implemented.

EXLEVEL_NS_EL1, bit [21]

Indicates if Non-secure EL1 is implemented.

EXLEVEL_NS_EL1	Meaning
0b0	Non-secure EL1 is not implemented.
0b1	Non-secure EL1 is implemented.

EXLEVEL_NS_EL0, bit [20]

Indicates if Non-secure EL0 is implemented.

EXLEVEL_NS_EL0	Meaning
0b0	Non-secure EL0 is not implemented.
0b1	Non-secure EL0 is implemented.

EXLEVEL_S_EL3, bit [19]

Indicates if EL3 is implemented.

EXLEVEL_S_EL3	Meaning
0b0	EL3 is not implemented.
0b1	EL3 is implemented.

EXLEVEL_S_EL2, bit [18]

Indicates if Secure EL2 is implemented.

EXLEVEL_S_EL2	Meaning
0b0	Secure EL2 is not implemented.
0b1	Secure EL2 is implemented.

EXLEVEL_S_EL1, bit [17]

Indicates if Secure EL1 is implemented.

EXLEVEL_S_EL1	Meaning
0b0	Secure EL1 is not implemented.
0b1	Secure EL1 is implemented.

EXLEVEL_S_EL0, bit [16]

Indicates if Secure EL0 is implemented.

EXLEVEL_S_EL0	Meaning
0b0	Secure EL0 is not implemented.
0b1	Secure EL0 is implemented.

Bits [15:14]

Reserved, RES0.

CCITMIN, bits [11:0]

Indicates the minimum value that can be programmed in [TRCCCCTLR.THRESHOLD](#).

If [TRCIDR0.TRCCCI](#) == 1 then the minimum value of this field is 0x001.

If [TRCIDR0.TRCCCI](#) == 0 then this field is zero.

Accessing TRCIDR3

TRCIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1EC	TRCIDR3

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCIDR4, ID Register 4

The TRCIDR4 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR4 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR4\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR4 are RES0.

Attributes

TRCIDR4 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMVMIDC	NUMCIDC	NUMSSCC	NUMRSPAIR	NUMPC	RES0	SUPPDAC	NUMDVC	NUMACPAIRS																							

NUMVMIDC, bits [31:28]

Indicates the number of Virtual Context Identifier Comparators that are available for tracing.

NUMVMIDC	Meaning
0b0000	No Virtual Context Identifier Comparators are available.
0b0001	The implementation has one Virtual Context Identifier Comparator.
0b0010	The implementation has two Virtual Context Identifier Comparators.
0b0011	The implementation has three Virtual Context Identifier Comparators.
0b0100	The implementation has four Virtual Context Identifier Comparators.
0b0101	The implementation has five Virtual Context Identifier Comparators.
0b0110	The implementation has six Virtual Context Identifier Comparators.
0b0111	The implementation has seven Virtual Context Identifier Comparators.
0b1000	The implementation has eight Virtual Context Identifier Comparators.

All other values are reserved.

NUMCIDC, bits [27:24]

Indicates the number of Context Identifier Comparators that are available for tracing.

NUMC IDC	Meaning
0b0000	No Context Identifier Comparators are available.
0b0001	The implementation has one Context Identifier Comparator.
0b0010	The implementation has two Context Identifier Comparators.
0b0011	The implementation has three Context Identifier Comparators.
0b0100	The implementation has four Context Identifier Comparators.
0b0101	The implementation has five Context Identifier Comparators.
0b0110	The implementation has six Context Identifier Comparators.
0b0111	The implementation has seven Context Identifier Comparators.
0b1000	The implementation has eight Context Identifier Comparators.

All other values are reserved.

NUMSSCC, bits [23:20]

Indicates the number of Single-shot Comparator Controls that are available for tracing.

NUMSSCC	Meaning
0b0000	No Single-shot Comparator Controls are available.
0b0001	The implementation has one Single-shot Comparator Control.
0b0010	The implementation has two Single-shot Comparator Controls.
0b0011	The implementation has three Single-shot Comparator Controls.
0b0100	The implementation has four Single-shot Comparator Controls.
0b0101	The implementation has five Single-shot Comparator Controls.
0b0110	The implementation has six Single-shot Comparator Controls.
0b0111	The implementation has seven Single-shot Comparator Controls.
0b1000	The implementation has eight Single-shot Comparator Controls.

All other values are reserved.

NUMRSPAIR, bits [19:16]

Indicates the number of resource selector pairs that are available for tracing.

NUMRSPAIR	Meaning
0b0000	The implementation has zero resource selectors.
0b0001	The implementation has two resource selector pairs.
0b0010	The implementation has three resource selector pairs.
0b0011	The implementation has four resource selector pairs.
0b0100	The implementation has five resource selector pairs.
0b0101	The implementation has six resource selector pairs.
0b0110	The implementation has seven resource selector pairs.
0b0111	The implementation has eight resource selector pairs.
0b1000	The implementation has nine resource selector pairs.
0b1001	The implementation has ten resource selector pairs.
0b1010	The implementation has eleven resource selector pairs.
0b1011	The implementation has twelve resource selector pairs.
0b1100	The implementation has thirteen resource selector pairs.
0b1101	The implementation has fourteen resource selector pairs.
0b1110	The implementation has fifteen resource selector pairs.
0b1111	The implementation has sixteen resource selector pairs.

All other values are reserved.

NUMPC, bits [15:12]

Indicates the number of PE Comparator Inputs that are available for tracing.

NUMPC	Meaning
0b0000	No PE Comparator Inputs are available.
0b0001	The implementation has one PE Comparator Input.
0b0010	The implementation has two PE Comparator Inputs.
0b0011	The implementation has three PE Comparator Inputs.
0b0100	The implementation has four PE Comparator Inputs.
0b0101	The implementation has five PE Comparator Inputs.
0b0110	The implementation has six PE Comparator Inputs.
0b0111	The implementation has seven PE Comparator Inputs.
0b1000	The implementation has eight PE Comparator Inputs.

All other values are reserved.

Bits [11:9]

Reserved, RES0.

SUPPDAC, bit [8]

When TRCIDR4.NUMACPAIRS != 0b0000:

Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

SUPPDAC	Meaning
0b0	Data address comparisons not implemented.
0b1	Data address comparisons implemented.

This field reads as 0.

Otherwise:

Reserved, RES0.

NUMDVC, bits [7:4]

Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

NUMDVC	Meaning
0b0000	No data value comparators implemented.
0b0001	One data value comparator implemented.
0b0010	Two data value comparators implemented.
0b0011	Three data value comparators implemented.
0b0100	Four data value comparators implemented.
0b0101	Five data value comparators implemented.
0b0110	Six data value comparators implemented.
0b0111	Seven data value comparators implemented.
0b1000	Eight data value comparators implemented.

All other values are reserved.

This field reads as 0b0000.

NUMACPAIRS, bits [3:0]

Indicates the number of Address Comparator pairs that are available for tracing.

NUMACPAIRS	Meaning
0b0000	No Address Comparator pairs are available.
0b0001	The implementation has one Address Comparator pair.
0b0010	The implementation has two Address Comparator pairs.
0b0011	The implementation has three Address Comparator pairs.
0b0100	The implementation has four Address Comparator pairs.
0b0101	The implementation has five Address Comparator pairs.
0b0110	The implementation has six Address Comparator pairs.
0b0111	The implementation has seven Address Comparator pairs.
0b1000	The implementation has eight Address Comparator pairs.

All other values are reserved.

Accessing TRCIDR4

TRCIDR4 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1F0	TRCIDR4

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCIDR5, ID Register 5

The TRCIDR5 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR5 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR5\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR5 are RES0.

Attributes

TRCIDR5 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OE	RES0	NUMCNTR	NUMSEQSTATE	RES0	LPOVERRIDE	ATBTRIG	TRACEIDSIZE	RES0	NUMEXTINSEL	NUMEXTIN																					

OE, bit [31]

Indicates support for the ETE Trace Output Enable.

Reserved, RES0.

OE	Meaning
0b0	ETE Trace Output Enable is not implemented.
0b1	ETE Trace Output Enable is implemented.

When FEAT_ETEv1p3 is implemented and when any IMPLEMENTATION DEFINED trace output interface is implemented, this field is 1.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

NUMCNTR, bits [30:28]

Indicates the number of Counters that are available for tracing.

NUMCNTR	Meaning
0b000	No Counters are available.
0b001	One Counter implemented.
0b010	Two Counters implemented.
0b011	Three Counters implemented.
0b100	Four Counters implemented.

All other values are reserved.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0b000.

NUMSEQSTATE, bits [27:25]

Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented.

NUMSEQSTATE	Meaning
0b000	The Sequencer is not implemented.
0b100	Four Sequencer states are implemented.

All other values are reserved.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0b000.

Bit [24]

Reserved, RES0.

LPOVERRIDE, bit [23]

Indicates support for Low-power Override Mode.

LPOVERRIDE	Meaning
0b0	The trace unit does not support Low-power Override Mode.
0b1	The trace unit supports Low-power Override Mode.

ATBTRIG, bit [22]

Indicates if the implementation can support ATB triggers.

ATBTRIG	Meaning
0b0	The implementation does not support ATB triggers.
0b1	The implementation supports ATB triggers.

If [TRCIDR4](#).NUMRSPAIR == 0b0000 then this field is 0.

TRACEIDSIZE, bits [21:16]

Indicates the trace ID width.

TRACEIDSIZE	Meaning
0b000000	The external trace interface is not implemented.
0b000111	The implementation supports a 7-bit trace ID.

All other values are reserved.

Note that AMBA ATB requires a 7-bit trace ID width.

Bits [15:12]

Reserved, RES0.

NUMEXTINSEL, bits [11:9]

Indicates how many External Input Selector resources are implemented.

NUMEXTINSEL	Meaning
0b000	No External Input Selector resources are available.
0b001	1 External Input Selector resource is available.
0b010	2 External Input Selector resources are available.
0b011	3 External Input Selector resources are available.
0b100	4 External Input Selector resources are available.

All other values are reserved.

NUMEXTIN, bits [8:0]

Indicates how many External Inputs are implemented.

NUMEXTIN	Meaning
0b11111111	Unified PMU event selection.

All other values are reserved.

Accessing TRCIDR5

TRCIDR5 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1F4	TRCIDR5

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCIDR6, ID Register 6

The TRCIDR6 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

External register TRCIDR6 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR6\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCIDR6 are RES0.

Attributes

TRCIDR6 is a 32-bit register.

Field descriptions

313029282726252423222120191817161514131211109876543	2	1	0
RES0	EXLEVEL_RL_EL2	EXLEVEL_RL_EL1	EXLEVEL_RL_EL0

Bits [31:3]

Reserved, RES0.

EXLEVEL_RL_EL2, bit [2]

Indicates if Realm EL2 is implemented.

EXLEVEL_RL_EL2	Meaning
0b0	Realm EL2 is not implemented.
0b1	Realm EL2 is implemented.

EXLEVEL_RL_EL1, bit [1]

Indicates if Realm EL1 is implemented.

EXLEVEL_RL_EL1	Meaning
0b0	Realm EL1 is not implemented.
0b1	Realm EL1 is implemented.

EXLEVEL_RL_EL0, bit [0]

Indicates if Realm EL0 is implemented.

EXLEVEL_RL_EL0	Meaning
0b0	Realm EL0 is not implemented.
0b1	Realm EL0 is implemented.

Accessing TRCIDR6

TRCIDR6 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1F8	TRCIDR6

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCIDR7, ID Register 7

The TRCIDR7 characteristics are:

Purpose

Returns the tracing capabilities of the trace unit.

Configuration

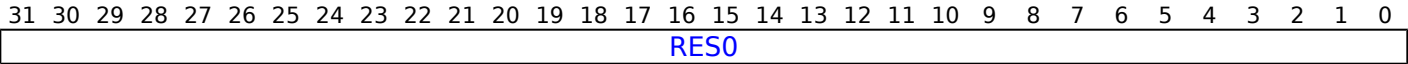
External register TRCIDR7 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR7\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCIDR7 are RES0.

Attributes

TRCIDR7 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCIDR7

TRCIDR7 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1FC	TRCIDR7

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

TRCIDR8, ID Register 8

The TRCIDR8 characteristics are:

Purpose

Returns the maximum speculation depth of the instruction trace element stream.

Configuration

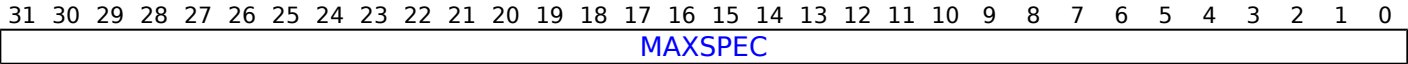
External register TRCIDR8 bits [31:0] are architecturally mapped to AArch64 System register [TRCIDR8\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCIDR8 are RES0.

Attributes

TRCIDR8 is a 32-bit register.

Field descriptions



MAXSPEC, bits [31:0]

Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time.

Accessing TRCIDR8

TRCIDR8 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x180	TRCIDR8

This interface is accessible as follows:

- When OSLockStatus() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCIMSPEC0, IMP DEF Register 0

The TRCIMSPEC0 characteristics are:

Purpose

TRCIMSPEC0 shows the presence of any IMPLEMENTATION DEFINED features, and provides an interface to enable the features that are provided.

Configuration

External register TRCIMSPEC0 bits [31:0] are architecturally mapped to AArch64 System register [TRCIMSPEC0\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented and FEAT_TRC_EXT is** implemented. Otherwise, direct accesses to TRCIMSPEC0 are RES0.

Attributes

TRCIMSPEC0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								EN		SUPPORT					

Bits [31:8]

Reserved, RES0.

EN, bits [7:4]

When TRCIMSPEC0.SUPPORT != 0b0000:

Enable. Controls whether the IMPLEMENTATION DEFINED features are enabled.

EN	Meaning
0b0000	The IMPLEMENTATION DEFINED features are not enabled. The trace unit must behave as if the IMPLEMENTATION DEFINED features are not supported.
0b0001	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0010	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0011	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0100	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0101	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0110	The trace unit behavior is IMPLEMENTATION DEFINED.
0b0111	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1000	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1001	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1010	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1011	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1100	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1101	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1110	The trace unit behavior is IMPLEMENTATION DEFINED.
0b1111	The trace unit behavior is IMPLEMENTATION DEFINED.

The reset behavior of this field is:

Reserved, RES0.

Indicates whether the implementation supports IMPLEMENTATION DEFINED features.

Use of nonzero values requires written permission from Arm.

Access to this field is **RO**.

TRCIMSPEC0 can be accessed through the external debug interface:

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

TRCIMSPEC<n>, IMP DEF Register <n>, n = 1 - 7

The TRCIMSPEC<n> characteristics are:

Purpose

These registers might return information that is specific to an implementation, or enable features specific to an implementation to be programmed. The product Technical Reference Manual describes these registers.

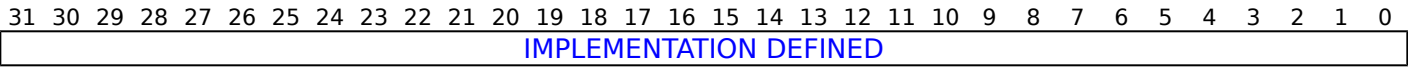
Configuration

External register TRCIMSPEC<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCIMSPEC<n>\[31:0\]](#).
This register is present only when the trace unit implements this OPTIONAL register, **FEAT_ETE is implemented** and **FEAT_TRC_EXT** ~~FEAT_ETE~~ is implemented. Otherwise, direct accesses to TRCIMSPEC<n> are RES0.

Attributes

TRCIMSPEC<n> is a 32-bit register.

Field descriptions



IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.
This field reads as an IMPLEMENTATION DEFINED value and writes to this field have IMPLEMENTATION DEFINED behavior.

Accessing TRCIMSPEC<n>

TRCIMSPEC<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x1C0 + (4 * n)	TRCIMSPEC<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

(old)

htmldiff from-

(new)

TRCITCTRL, Integration Mode Control Register

The TRCITCTRL characteristics are:

Purpose

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCITCTRL are RES0.

Attributes

TRCITCTRL is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																IME															

Bits [31:1]

Reserved, RES0.

IME, bit [0] When topology detection or integration functionality is implemented:

Integration Mode Enable.

IME	Meaning
0b0	Component functional mode.
0b1	Component integration mode. Support for topology detection and integration testing is enabled.

Otherwise:

Reserved, RES0.

Accessing TRCITCTRL

External debugger accesses to this register are IMPLEMENTATION DEFINED when the trace unit is not in the Idle state.

TRCITCTRL can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

TRCITCTRL, Integration Mode Control Register

ETE	0xF00	TRCITCTRL
-----	-------	-----------

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCITEEDCR, Instrumentation Trace Extension External Debug Control Register

The TRCITEEDCR characteristics are:

Purpose

Controls instrumentation trace filtering.

Configuration

This register is present only when FEAT_ETE is implemented, FEAT_TRC_EXT is implemented and TRCIDR0.ITE == 1. Otherwise, direct accesses to TRCITEEDCR are RES0.

Attributes

TRCITEEDCR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								RL	S	NS	E3	E2	E1	E0	

Bits [31:7]

Reserved, RES0.

RL, bit [6] When FEAT_RME is implemented:

Instrumentation Trace in Realm state.

RL	Meaning
0b0	Instrumentation trace prohibited in Realm state.
0b1	Instrumentation trace permitted in Realm state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Realm state.

This field is ignored when SelfHostedTraceEnabled() returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

S, bit [5]**When Secure state is implemented:**

Instrumentation Trace in Secure state.

S	Meaning
0b0	Instrumentation trace prohibited in Secure state.
0b1	Instrumentation trace permitted in Secure state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Secure state.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NS, bit [4]**When Any of Non-secure EL2, EL1, or EL0 are implemented:**

Instrumentation Trace in Non-secure state.

NS	Meaning
0b0	Instrumentation trace prohibited in Non-secure state.
0b1	Instrumentation trace permitted in Non-secure state.

Used in conjunction with [TRCCONFIGR.ITO](#) and [TRCITEEDCR.E<m>](#) to control whether Instrumentation trace is permitted or prohibited in Non-secure state.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E3, bit [3]**When EL3 is implemented:**

Instrumentation Trace Enable at EL3.

E3	Meaning
0b0	Instrumentation trace prohibited at EL3.
0b1	Instrumentation trace permitted at EL3.

This field is ignored when `SelfHostedTraceEnabled()` returns TRUE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

E<m>, bit [m], for m = 2 to 0

Instrumentation Trace Enable at EL<m>.

E<m>	Meaning
0b0	Instrumentation trace prohibited at EL<m>.
0b1	Instrumentation trace permitted at EL<m>.

Used in conjunction with [TRCCONFIGR.ITO](#), [TRCITEEDCR.NS](#), [TRCITEEDCR.S](#), and [TRCITEEDCR.RL](#) to control whether Instrumentation trace is permitted or prohibited at EL<m> in the specified security states.

This field is ignored when SelfHostedTraceEnabled() returns TRUE.

E<2> is RES0 if EL2 is not implemented in any security states.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCITEEDCR

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCITEEDCR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x048	TRCITEEDCR

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:58; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file htmldiff from- (new)

(old)

htmldiff from-

(new)

TRCLAR, Lock Access Register

The TRCLAR characteristics are:

Purpose

Used to lock and unlock the Software Lock.

Note that ETE does not implement the Software Lock.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and the Software Lock is implemented. Otherwise, direct accesses to TRCLAR are RES0.

Attributes

TRCLAR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																KEY															

KEY, bits [31:0]

When Software Lock is implemented:

Software Lock Key.

A value of 0xC5ACCE55 unlocks the Software Lock.

Any other value locks the Software Lock.

Otherwise:

Reserved, RES0.

Accessing TRCLAR

External debugger accesses to this register are unaffected by the OS Lock.

TRCLAR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFB0	TRCLAR

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **WO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCLSR, Lock Status Register

The TRCLSR characteristics are:

Purpose

Indicates whether the Software Lock is implemented, and the current status of the Software Lock.
For additional information, see the CoreSight Architecture Specification.

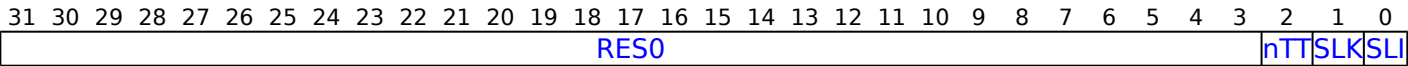
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCLSR are RES0.

Attributes

TRCLSR is a 32-bit register.

Field descriptions



Bits [31:3]

Reserved, RES0.

nTT, bit [2]

Software lock size.
Reads as 0b0.
Access to this field is **RO**.

SLK, bit [1]

The current Software Lock status.

SLK	Meaning
0b0	Software Lock is unlocked.
0b1	Software Lock is locked. Writes to the other registers in this component, except for the TRCLAR , are ignored.

This field reads as 0.

SLI, bit [0]

Indicates whether the Software Lock is implemented.

SLI	Meaning
0b0	Software Lock is not implemented. Writes to the TRCLAR are ignored.
0b1	Software Lock is implemented.

This field reads as 0.

Accessing TRCLSR

External debugger accesses to this register are unaffected by the OS Lock.

TRCLSR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFB4	TRCLSR

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCOSLSR characteristics are:

Purpose

Returns the status of the Trace OS Lock.

Configuration

External register TRCOSLSR bits [31:0] are architecturally mapped to AArch64 System register [TRCOSLSR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCOSLSR are RES0.

Attributes

TRCOSLSR is a 32-bit register.

Field descriptions

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

RES0 OSLM[2:1] RES0 OSLK OSLM[0]

Bits [31:5]

Reserved, RES0.

OSLM, bits [4:3, 0]

OS Lock model.

OSLM	Meaning
0b000	Trace OS Lock is not implemented.
0b010	Trace OS Lock is implemented.
0b100	Trace OS Lock is not implemented, and the trace unit is controlled by the PE OS Lock.

All other values are reserved.

This field reads as 0b100.

The OSLM field is split as follows:

- OSLM[2:1] is TRCOSLSR[4:3].
- OSLM[0] is TRCOSLSR[0].

Bit [2]

Reserved, RES0.

OSLK, bit [1]

OS Lock status.

OSLK	Meaning
0b0	The OS Lock is unlocked.
0b1	The OS Lock is locked.

Note that this field indicates the state of the PE OS Lock.

Accessing TRCOSLSR

External debugger accesses to this register are unaffected by the OS Lock.

TRCOSLSR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x304	TRCOSLSR

This interface is accessible as follows:

- When `!AllowExternalTraceAccess()` or `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCPDCR, PowerDown Control Register

The TRCPDCR characteristics are:

Purpose

Requests the system to provide power to the trace unit.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPDCR are RES0.

Attributes

TRCPDCR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																													PU	RES0	

Bits [31:4]

Reserved, RES0.

PU, bit [3]

Power Up Request.

PU	Meaning
0b0	The system can remove power from the trace unit core power domain, or requests for power to the trace unit core power domain are implemented outside of the trace unit.
0b1	The system must provide power to the trace unit core power domain.

This field is RES0.

Bits [2:0]

Reserved, RES0.

Accessing TRCPDCR

External debugger accesses to this register are unaffected by the OS Lock.

TRCPDCR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x310	TRCPDCR

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808: 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCPDSR, PowerDown Status Register

The TRCPDSR characteristics are:

Purpose

Indicates the power status of the trace unit.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPDSR are RES0.

Attributes

TRCPDSR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0												OSLK		RES0		STICKYPD		POWER	

Bits [31:6]

Reserved, RES0.

OSLK, bit [5]

OS Lock Status.

OSLK	Meaning
0b0	The OS Lock is unlocked.
0b1	The OS Lock is locked.

Note that this field indicates the state of the PE OS Lock.

Bits [4:2]

Reserved, RES0.

STICKYPD, bit [1]

Sticky powerdown status. Indicates whether the trace register state is valid.

STICKYPD	Meaning
0b0	The state of TRCOSLSR and the trace registers are valid.
0b1	The state of TRCOSLSR and the trace registers might not be valid.

This field is set to 1 if the power to the trace unit core power domain is removed and the trace unit register state is not valid.

The STICKYPD field is read-sensitive. On a read of the TRCPDSR, this field is cleared to 0 after the register has been read.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 1.

POWER, bit [0]

Power Status.

POWER	Meaning
0b0	The trace unit core power domain is not powered. All trace unit registers are not accessible and they all return an error response.
0b1	The trace unit core power domain is powered. Trace unit registers are accessible.

Access to this field is **RAO/WI**.

Accessing TRCPDSR

External debugger accesses to this register are unaffected by the OS Lock.

TRCPDSR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x314	TRCPDSR

This interface is accessible as follows:

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCPIDR0, Peripheral Identification Register 0

The TRCPIDR0 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCPIDR0 are RES0.

Attributes

TRCPIDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PART_0							

Bits [31:8]

Reserved, RES0.

PART_0, bits [7:0]

Part number, bits [7:0].

The part number is selected by the designer of the component, and is stored in [TRCPIDR1](#).PART_1 and TRCPIDR0.PART_0.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRCPIDR0

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFE0	TRCPIDR0

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCPIDR1, Peripheral Identification Register 1

The TRCPIDR1 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPIDR1 are RES0.

Attributes

TRCPIDR1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								DES 0			PART 1				

Bits [31:8]

Reserved, RES0.

DES_0, bits [7:4]

Designer, JEP106 identification code, bits [3:0]. TRCPIDR1.DES_0 and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 identification code is 0x3B.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PART_1, bits [3:0]

Part number, bits [11:8].

The part number is selected by the designer of the component, and is stored in TRCPIDR1.PART_1 and TRCPIDR0.PART_0.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

TRCPIDR1 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFE4	TRCPIDR1

- When `!IsTraceCorePowered()`, accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCPIDR2, Peripheral Identification Register 2

The TRCPIDR2 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCPIDR2 are RES0.

Attributes

TRCPIDR2 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																REVISION				JEDEC		DES 1									

Bits [31:8]

Reserved, RES0.

REVISION, bits [7:4]

Component major revision. TRCPIDR2.REVISION and [TRCPIDR3.REVAND](#) together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and [TRCPIDR3.REVAND](#) the least significant part. When a component is changed, TRCPIDR2.REVISION or [TRCPIDR3.REVAND](#) are increased to ensure that software can differentiate the different revisions of the component. [TRCPIDR3.REVAND](#) should be set to 0b0000 when TRCPIDR2.REVISION is increased.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

JEDEC, bit [3]

JEDEC-assigned JEP106 implementer code is used.

Reads as 0b1.

Access to this field is **RO**.

DES_1, bits [2:0]

Designer, JEP106 identification code, bits [6:4]. [TRCPIDR1.DES_0](#) and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 identification code is 0x3B.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRCPIDR2

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFE8	TRCPIDR2

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCPIDR3, Peripheral Identification Register 3

The TRCPIDR3 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCPIDR3 are RES0.

Attributes

TRCPIDR3 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0												REVAND				CMOD			

Bits [31:8]

Reserved, RES0.

REVAND, bits [7:4]

Component minor revision. [TRCPIDR2.REVISION](#) and TRCPIDR3.REVAND together form the revision number of the component, with [TRCPIDR2.REVISION](#) being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, [TRCPIDR2.REVISION](#) or TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. TRCPIDR3.REVAND should be set to 0b0000 when [TRCPIDR2.REVISION](#) is increased.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

CMOD, bits [3:0]

Customer Modified.

Indicates the component has been modified.

A value of 0b0000 means the component is not modified from the original design.

Any other value means the component has been modified in an IMPLEMENTATION DEFINED way.

For any two components with the same Unique Component Identifier:

- If the value of the CMOD fields of both components equals zero, the components are identical.
- If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have the same modifications.

- If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same Unique Component Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRCPIDR3

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFEC	TRCPIDR3

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCPIDR4, Peripheral Identification Register 4

The TRCPIDR4 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPIDR4 are RES0.

Attributes

TRCPIDR4 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								SIZE			DES 2				

Bits [31:8]

Reserved, RES0.

SIZE, bits [7:4]

Size of the component.

The distance from the start of the address space used by this component to the end of the component identification registers.

A value of 0b0000 means one of the following is true:

- The component uses a single 4KB block.
- The component uses an IMPLEMENTATION DEFINED number of 4KB blocks.

Any other value means the component occupies $2^{\text{TRCPIDR4.SIZE}}$ 4KB blocks.

Using this field to indicate the size of the component is deprecated. This field might not correctly indicate the size of the component. Arm recommends that software determine the size of the component from the Unique Component Identifier fields, and other IMPLEMENTATION DEFINED registers in the component.

Reads as 0b0000.

Access to this field is **RO**.

DES_2, bits [3:0]

Designer, JEP106 continuation code. This is the JEDEC-assigned JEP106 bank identifier for the designer of the component, minus 1. The code identifies the designer of the component, which might not be the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC <http://www.jedec.org>.

Note

For a component designed by Arm Limited, the JEP106 bank is 5, meaning this field has the value 0x4.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing TRCPIDR4

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR4 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFD0	TRCPIDR4

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCPIDR5, Peripheral Identification Register 5

The TRCPIDR5 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

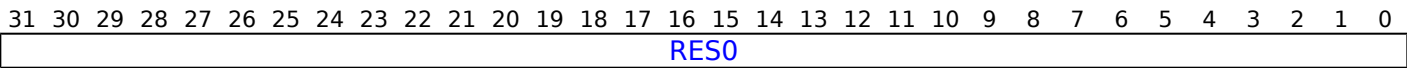
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPIDR5 are RES0.

Attributes

TRCPIDR5 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCPIDR5

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR5 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFD4	TRCPIDR5

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

TRCPIDR6, Peripheral Identification Register 6

The TRCPIDR6 characteristics are:

Purpose

Provides discovery information about the component.
For additional information, see the CoreSight Architecture Specification.

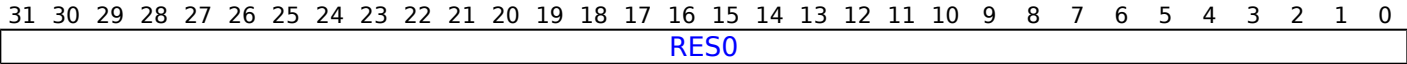
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPIDR6 are RES0.

Attributes

TRCPIDR6 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCPIDR6

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR6 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFD8	TRCPIDR6

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

TRCPIDR7, Peripheral Identification Register 7

The TRCPIDR7 characteristics are:

Purpose

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

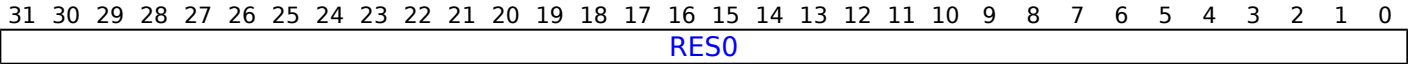
Configuration

This register is present only when FEAT_ETE is implemented and FEAT_TRC_EXT is implemented. Otherwise, direct accesses to TRCPIDR7 are RES0.

Attributes

TRCPIDR7 is a 32-bit register.

Field descriptions



Bits [31:0]

Reserved, RES0.

Accessing TRCPIDR7

External debugger accesses to this register are unaffected by the OS Lock.

TRCPIDR7 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0xFDC	TRCPIDR7

This interface is accessible as follows:

- When !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

TRCPRGCTLR, Programming Control Register

The TRCPRGCTLR characteristics are:

Purpose

Enables the trace unit.

Configuration

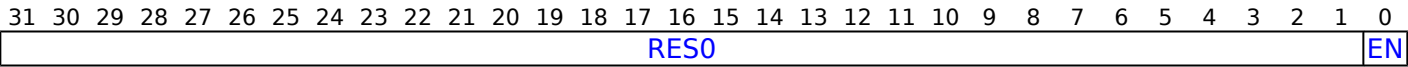
External register TRCPRGCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCPRGCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented and FEAT_TRC_EXT is** implemented. Otherwise, direct accesses to TRCPRGCTLR are RES0.

Attributes

TRCPRGCTLR is a 32-bit register.

Field descriptions



Bits [31:1]

Reserved, RES0.

EN, bit [0]

Trace unit enable.

EN	Meaning
0b0	The trace unit is disabled.
0b1	The trace unit is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to 0.

Accessing TRCPRGCTLR

Must be programmed.

TRCPRGCTLR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x004	TRCPRGCTLR

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.

- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCQCTLR, Q Element Control Register

The TRCQCTLR characteristics are:

Purpose

Controls when Q elements are enabled.

Configuration

External register TRCQCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCQCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR0.QFILT == 1. Otherwise, direct accesses to TRCQCTLR are RES0.

Attributes

TRCQCTLR is a 32-bit register.

Field descriptions

313029282726252423222120191817161514131211109	8	7	6	5	4	3	2
RES0	MODE	RANGE[7]	RANGE[6]	RANGE[5]	RANGE[4]	RANGE[3]	RANGE[2]

Bits [31:9]

Reserved, RES0.

MODE, bit [8]

Selects whether the Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit is permitted to generate Q elements or address ranges where the trace unit is not permitted to generate Q elements:

MODE	Meaning
0b0	Exclude mode. The Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit must not generate Q elements. If no ranges are selected, Q elements are permitted across the entire memory map.
0b1	Include Mode. The Address Range Comparators selected by TRCQCTLR.RANGE indicate address ranges where the trace unit can generate Q elements. If all the implemented bits in RANGE are set to 0 then Q elements are disabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

RANGE[<m>], bit [m], for m = 7 to 0

Specifies whether Address Range Comparator <m> controls Q elements.

RANGE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected.
0b1	The address range that Address Range Comparator <m> defines, is selected.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCQCTLR

Must be programmed if [TRCCONFIGR.QE](#) != 0b00.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCQCTLR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x044	TRCQCTL

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCRSCTLR<n>, Resource Selection Control Register <n>, n = 2 - 31

The TRCRSCTLR<n> characteristics are:

Purpose

Controls the selection of the resources in the trace unit.

Configuration

External register TRCRSCTLR<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCRSCTLR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $(\text{UInt}(\text{TRCIDR4.NUMRSPAIR}) + 1) * 2 > n$. Otherwise, direct accesses to TRCRSCTLR<n> are RES0.

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

TRCRSCTLR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0										PAIRINV	INV	GROUP				SELECT															

Bits [31:22]

Reserved, RES0.

PAIRINV, bit [21]

When $n \bmod 2 == 0$:

Controls whether the combined result from a resource selector pair is inverted.

PAIRINV	Meaning
0b0	Do not invert the combined output of the 2 resource selectors.
0b1	Invert the combined output of the 2 resource selectors.

If:

- A is the register TRCRSCTLR<n>.
- B is the register TRCRSCTLR<n+1>.

Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows:

- 0b000 -> A and B.

- 0b001 -> Reserved.
- 0b010 -> not(A) and B.
- 0b011 -> not(A) and not(B).
- 0b100 -> not(A) or not(B).
- 0b101 -> not(A) or B.
- 0b110 -> Reserved.
- 0b111 -> A or B.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

INV, bit [20]

Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted.

INV	Meaning
0b0	Do not invert the output of this selector.
0b1	Invert the output of this selector.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

GROUP, bits [19:16]

Selects a group of resources.

GROUP	Meaning	SELECT
0b0000	External Input Selectors.	SELECT encoding for External Input Selectors
0b0001	PE Comparator Inputs.	SELECT encoding for PE Comparator Inputs
0b0010	Counters and Sequencer.	SELECT encoding for Counters and Sequencer
0b0011	Single-shot Comparator Controls.	SELECT encoding for Single-shot Comparator Controls
0b0100	Single Address Comparators.	SELECT encoding for Single Address Comparators
0b0101	Address Range Comparators.	SELECT encoding for Address Range Comparators
0b0110	Context Identifier Comparators.	SELECT encoding for Context Identifier Comparators
0b0111	Virtual Context Identifier Comparators.	SELECT encoding for Virtual Context Identifier Comparators

All other values are reserved.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT, bits [15:0]

Resource Specific Controls. Contains the controls specific to the resource group selected by GROUP, described in the following sections.

SELECT encoding for External Input Selectors

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												EXTIN[3]	EXTIN[2]	EXTIN[1]	EXTIN[0]

Bits [15:4]

Reserved, RES0.

EXTIN[<m>], bit [m], for m = 3 to 0

Selects one or more External Inputs.

EXTIN[<m>]	Meaning
0b0	Ignore EXTIN <m>.
0b1	Select EXTIN <m>.

This bit is RES0 if m >= [TRCIDR5](#).NUMEXTINSEL.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for PE Comparator Inputs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								PECOMP[7]	PECOMP[6]	PECOMP[5]	PECOMP[4]	PECOMP[3]	PECOMP[2]	PECOMP[1]	PECOMP[0]

Bits [15:8]

Reserved, RES0.

PECOMP[<m>], bit [m], for m = 7 to 0

Selects one or more PE Comparator Inputs.

PECOMP[<m>]	Meaning
0b0	Ignore PE Comparator Input <m>.
0b1	Select PE Comparator Input <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMPC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Counters and Sequencer

15	14	13	12	11	10	9	8	7	6	5	4	3	2
RES0								SEQUENCER[3]	SEQUENCER[2]	SEQUENCER[1]	SEQUENCER[0]	COUNTERS[3]	COUNTERS[2]

Bits [15:8]

Reserved, RES0.

SEQUENCER[<m>], bit [m+4], for m = 3 to 0

Sequencer states.

SEQUENCER[<m>]	Meaning
0b0	Ignore Sequencer state <m>.
0b1	Select Sequencer state <m>.

This bit is RES0 if m >= [TRCIDR5](#).NUMSEQSTATE.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

COUNTERS[<m>], bit [m], for m = 3 to 0

Counters resources at zero.

COUNTERS[<m>]	Meaning
0b0	Ignore Counter <m>.
0b1	Select Counter <m> is zero.

This bit is RES0 if m >= [TRCIDR5](#).NUMCNTR.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Single-shot Comparator Controls

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	SINGLE_SHOT[7]	SINGLE_SHOT[6]	SINGLE_SHOT[5]	SINGLE_SHOT[4]	SINGLE_SHOT[3]	SINGLE_SHOT[2]	SINGLE_SHOT[1]	SINGLE_SHOT[0]	RES0	RES0	RES0	RES0	RES0	RES0	RES0

Bits [15:8]

Reserved, RES0.

SINGLE_SHOT[<m>], bit [m], for m = 7 to 0

Selects one or more Single-shot Comparator Controls.

SINGLE_SHOT[<m>]	Meaning
0b0	Ignore Single-shot Comparator Control <m>.
0b1	Select Single-shot Comparator Control <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMSSCC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Single Address Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAC[15]	SAC[14]	SAC[13]	SAC[12]	SAC[11]	SAC[10]	SAC[9]	SAC[8]	SAC[7]	SAC[6]	SAC[5]	SAC[4]	SAC[3]	SAC[2]	SAC[1]	SAC[0]

SAC[<m>], bit [m], for m = 15 to 0

Selects one or more Single Address Comparators.

SAC[<m>]	Meaning
0b0	Ignore Single Address Comparator <m>.
0b1	Select Single Address Comparator <m>.

This bit is RES0 if m >= 2 × [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Address Range Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								ARC[7]	ARC[6]	ARC[5]	ARC[4]	ARC[3]	ARC[2]	ARC[1]	ARC[0]

Bits [15:8]

Reserved, RES0.

ARC[<m>], bit [m], for m = 7 to 0

Selects one or more Address Range Comparators.

ARC[<m>]	Meaning
0b0	Ignore Address Range Comparator <m>.
0b1	Select Address Range Comparator <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Context Identifier Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								CID[7]	CID[6]	CID[5]	CID[4]	CID[3]	CID[2]	CID[1]	CID[0]

Bits [15:8]

Reserved, RES0.

CID[<m>], bit [m], for m = 7 to 0

Selects one or more Context Identifier Comparators.

CID[<m>]	Meaning
0b0	Ignore Context Identifier Comparator <m>.
0b1	Select Context Identifier Comparator <m>.

This bit is RES0 if m >= [TRCIDR4](#).NUMCIDC.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SELECT encoding for Virtual Context Identifier Comparators

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								VMID[7]	VMID[6]	VMID[5]	VMID[4]	VMID[3]	VMID[2]	VMID[1]	VMID[0]

Bits [15:8]

Reserved, RES0.

VMID[<m>], bit [m], for m = 7 to 0

Selects one or more Virtual Context Identifier Comparators.

VMID[<m>]	Meaning
0b0	Ignore Virtual Context Identifier Comparator <m>.
0b1	Select Virtual Context Identifier Comparator <m>.

This bit is RES0 if m >= [TRCIDR4.NUMVMIDC](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCRSCTLR<n>

Must be programmed if any of the following are true:

- [TRCCNTCTLR<a>.RLDEVENT.TYPE](#) == 0 and [TRCCNTCTLR<a>.RLDEVENT.SEL](#) == n.
- [TRCCNTCTLR<a>.RLDEVENT.TYPE](#) == 1 and [TRCCNTCTLR<a>.RLDEVENT.SEL](#) == n/2.
- [TRCCNTCTLR<a>.CNTEVENT.TYPE](#) == 0 and [TRCCNTCTLR<a>.CNTEVENT.SEL](#) == n.
- [TRCCNTCTLR<a>.CNTEVENT.TYPE](#) == 1 and [TRCCNTCTLR<a>.CNTEVENT.SEL](#) == n/2.
- [TRCEVENTCTL0R.EVENT0.TYPE](#) == 0 and [TRCEVENTCTL0R.EVENT0.SEL](#) == n.
- [TRCEVENTCTL0R.EVENT0.TYPE](#) == 1 and [TRCEVENTCTL0R.EVENT0.SEL](#) == n/2.
- [TRCEVENTCTL0R.EVENT1.TYPE](#) == 0 and [TRCEVENTCTL0R.EVENT1.SEL](#) == n.
- [TRCEVENTCTL0R.EVENT1.TYPE](#) == 1 and [TRCEVENTCTL0R.EVENT1.SEL](#) == n/2.
- [TRCEVENTCTL0R.EVENT2.TYPE](#) == 0 and [TRCEVENTCTL0R.EVENT2.SEL](#) == n.
- [TRCEVENTCTL0R.EVENT2.TYPE](#) == 1 and [TRCEVENTCTL0R.EVENT2.SEL](#) == n/2.
- [TRCEVENTCTL0R.EVENT3.TYPE](#) == 0 and [TRCEVENTCTL0R.EVENT3.SEL](#) == n.
- [TRCEVENTCTL0R.EVENT3.TYPE](#) == 1 and [TRCEVENTCTL0R.EVENT3.SEL](#) == n/2.
- [TRCSEQEVR<a>.B.TYPE](#) == 0 and [TRCSEQEVR<a>.B.SEL](#) == n.
- [TRCSEQEVR<a>.B.TYPE](#) == 1 and [TRCSEQEVR<a>.B.SEL](#) == n/2.
- [TRCSEQEVR<a>.F.TYPE](#) == 0 and [TRCSEQEVR<a>.F.SEL](#) == n.
- [TRCSEQEVR<a>.F.TYPE](#) == 1 and [TRCSEQEVR<a>.F.SEL](#) == n/2.
- [TRCSEQRSTEV.RST.TYPE](#) == 0 and [TRCSEQRSTEV.RST.SEL](#) == n.
- [TRCSEQRSTEV.RST.TYPE](#) == 1 and [TRCSEQRSTEV.RST.SEL](#) == n/2.
- [TRCTSCTLR.EVENT.TYPE](#) == 0 and [TRCTSCTLR.EVENT.SEL](#) == n.
- [TRCTSCTLR.EVENT.TYPE](#) == 1 and [TRCTSCTLR.EVENT.SEL](#) == n/2.
- [TRCVICTLR.EVENT.TYPE](#) == 0 and [TRCVICTLR.EVENT.SEL](#) == n.
- [TRCVICTLR.EVENT.TYPE](#) == 1 and [TRCVICTLR.EVENT.SEL](#) == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCRSCTLR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x200 + (4 * n)	TRCRSCTLR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCRSR, Resources Status Register

The TRCRSR characteristics are:

Purpose

Use this to set, or read, the status of the resources.

Configuration

External register TRCRSR bits [31:0] are architecturally mapped to AArch64 System register [TRCRSR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCRSR are RES0.

Attributes

TRCRSR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0												TA	EVENT[3]	EVENT[2]	EVENT[1]	EVENT[0]	RES0	EXTIN[3]	EXTIN[2]	EXTIN[1]	EXTIN[0]										

Bits [31:13]

Reserved, RES0.

TA, bit [12]

Tracing active.

TA	Meaning
0b0	Tracing is not active.
0b1	Tracing is active.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

EVENT[<m>], bit [m+8], for m = 3 to 0

Untraced status of ETEEvents.

EVENT[<m>]	Meaning
0b0	An ETEEvent <m> has not occurred.
0b1	An ETEEvent <m> has occurred while the resources were in the Paused state.

This bit is RES0 if [TRCIDR4](#).NUMRSPAIR == 0 || m > [TRCIDR0](#).NUMEVENT.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

(old)

htmldiff from-

(new)

TRCSEQEVR<n>, Sequencer State Transition Control Register <n>, n = 0 - 2

The TRCSEQEVR<n> characteristics are:

Purpose

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configuration

External register TRCSEQEVR<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCSEQEVR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQEVR<n> are RES0.

Attributes

TRCSEQEVR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																B TYPE		RES0		B SEL				F TYPE		RES0		F SEL			

Bits [31:16]

Reserved, RES0.

B_TYPE, bit [15]

Chooses the type of Resource Selector.

Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == 0x14 then when event 0x14 occurs, the Sequencer moves from state 3 to state 2.

B_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [14:13]

Reserved, RES0.

B_SEL, bits [12:8]

Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Backward field. Selects the single Resource Selector or Resource Selector pair.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

F_TYPE, bit [7]

Chooses the type of Resource Selector.

Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2.

F_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

F_SEL, bits [4:0]

Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.

Forward field. Selects the single Resource Selector or Resource Selector pair.

If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

Selecting Resource Selector pair 0 using this field is UNPREDICTABLE, and the resource event might fire or might not fire when the resources are not in the Paused state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSEQEVR<n>

Must be programmed if [TRCRSCTLR<a>.GROUP == 0b0010](#) and [TRCRSCTLR<a>.SEQUENCER != 0b0000](#).

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCSEQEVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x100 + (4 * n)	TRCSEQEVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSEQRSTEV, Sequencer Reset Control Register

The TRCSEQRSTEV characteristics are:

Purpose

Moves the Sequencer to state 0 when a programmed resource event occurs.

Configuration

External register TRCSEQRSTEV bits [31:0] are architecturally mapped to AArch64 System register [TRCSEQRSTEV\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQRSTEV are RES0.

Attributes

TRCSEQRSTEV is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8		7		6	5	4	3	2	1	0
RES0																								RST_TYPE		RES0		RST_SEL					

Bits [31:8]

Reserved, RES0.

RST_TYPE, bit [7]

Chooses the type of Resource Selector.

RST_TYPE	Meaning
0b0	A single Resource Selector. TRCSEQRSTEV.RST.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCSEQRSTEV.RST.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQRSTEV.RST.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [6:5]

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCSEQSTR, Sequencer State Register

The TRCSEQSTR characteristics are:

Purpose

Use this to set, or read, the Sequencer state.

Configuration

External register TRCSEQSTR bits [31:0] are architecturally mapped to AArch64 System register [TRCSEQSTR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR5.NUMSEQSTATE != 0b000. Otherwise, direct accesses to TRCSEQSTR are RES0.

Attributes

TRCSEQSTR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															STATE

Bits [31:2]

Reserved, RES0.

STATE, bits [1:0]

Set or returns the state of the Sequencer.

STATE	Meaning
0b00	State 0.
0b01	State 1.
0b10	State 2.
0b11	State 3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSEQSTR

Must be programmed if [TRCRSCTLR<a>.GROUP](#) == 0b0010 and [TRCRSCTLR<a>.SEQUENCER](#) != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

TRCSEQSTR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x11C	TRCSEQSTR

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSSCCR<n>, Single-shot Comparator Control Register <n>, n = 0 - 7

The TRCSSCCR<n> characteristics are:

Purpose

Controls the corresponding Single-shot Comparator Control resource.

Configuration

External register TRCSSCCR<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCSSCCR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and UInt(TRCIDR4.NUMSSCC) > n. Otherwise, direct accesses to TRCSSCCR<n> are RES0.

Attributes

TRCSSCCR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	
							RES0	RST	ARC[7]	ARC[6]	ARC[5]	ARC[4]	ARC[3]	ARC[2]	ARC[1]	ARC[0]	SAC[15]	SAC[14]	SAC[13]	SAC[12]	SAC[11]

Bits [31:25]

Reserved, RES0.

RST, bit [24]

Selects the Single-shot Comparator Control mode.

RST	Meaning
0b0	The Single-shot Comparator Control is in single-shot mode.
0b1	The Single-shot Comparator Control is in multi-shot mode.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

ARC[<m>], bit [m+16], for m = 7 to 0

Selects one or more Address Range Comparators for Single-shot control.

ARC[<m>]	Meaning
0b0	The Address Range Comparator <m>, is not selected for Single-shot control.
0b1	The Address Range Comparator <m>, is selected for Single-shot control.

This bit is RES0 if m >= [TRCIDR4.NUMACPAIRS](#).

The reset behavior of this field is:

Selects one or more Single Address Comparators for Single-shot control.

SAC[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected for Single-shot control.
0b1	The Single Address Comparator <m>, is selected for Single-shot control.

This bit is RES0 if $m \geq 2 \times \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSSCCR<n>

Must be programmed if any [TRCRSCTLR<a>](#).GROUP == 0b0011 and [TRCRSCTLR<a>](#).SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCSSCCR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	$0x280 + (4 * n)$	TRCSSCCR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

TRCSSCSR<n>, Single-shot Comparator Control Status Register <n>, n = 0 - 7

The TRCSSCSR<n> characteristics are:

Purpose

Returns the status of the corresponding Single-shot Comparator Control.

Configuration

External register TRCSSCSR<n> bits [31:0] are architecturally mapped to AArch64 System register [TRCSSCSR<n>\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR4.NUMSSCC) > n$. Otherwise, direct accesses to TRCSSCSR<n> are RES0.

Attributes

TRCSSCSR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
STATUS		PENDING																										RES0								PCD		VD		DA		INST	

STATUS, bit [31]

Single-shot Comparator Control status. Indicates if any of the comparators selected by this Single-shot Comparator control have matched. The selected comparators are defined by [TRCSSCCR<n> .ARC](#), [TRCSSCCR<n> .SAC](#), and [TRCSSPCICR<n> .PC](#).

STATUS	Meaning
0b0	No match has occurred. When the first match occurs, this field takes a value of 1. It remains at 1 until explicitly modified by a write to this register.
0b1	One or more matches has occurred. If TRCSSCCR<n> .RST == 0 then: <ul style="list-style-type: none"> There is only one match and no more matches are possible. Software must reset this field to 0 to re-enable the Single-shot Comparator Control.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

PENDING, bit [30]

Single-shot pending status. The Single-shot Comparator Control fired while the resources were in the Paused state.

PENDING	Meaning
0b0	No match has occurred.
0b1	One or more matches has occurred.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [29:4]

Reserved, RES0.

PC, bit [3]

PE Comparator Input support. Indicates if the Single-shot Comparator Control supports PE Comparator Inputs.

PC	Meaning
0b0	This Single-shot Comparator Control does not support PE Comparator Inputs. Selecting any PE Comparator Inputs using the associated TRCSSPCICR<n> results in CONSTRAINED UNPREDICTABLE behavior of the Single-shot Comparator Control resource. The Single-shot Comparator Control might match unexpectedly or might not match.
0b1	This Single-shot Comparator Control supports PE Comparator Inputs.

Access to this field is **RO**.

DV, bit [2]

Data value comparator support. Data value comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

DV	Meaning
0b0	This Single-shot Comparator Control does not support data value comparisons.
0b1	This Single-shot Comparator Control supports data value comparisons.

This field reads as 0.

Access to this field is **RO**.

DA, bit [1]

Data Address Comparator support. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.

DA	Meaning
0b0	This Single-shot Comparator Control does not support data address comparisons.
0b1	This Single-shot Comparator Control supports data address comparisons.

This field reads as 0.

Access to this field is **RO**.

INST, bit [0]

Instruction Address Comparator support. Indicates if the Single-shot Comparator Control supports instruction address comparisons.

INST	Meaning
0b0	This Single-shot Comparator Control does not support instruction address comparisons.
0b1	This Single-shot Comparator Control supports instruction address comparisons.

TRCSSPCICR<n>, Single-shot Processing Element Comparator Input Control Register <n>, n = 0 - 7

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

TRCSSPCICR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x2C0 + (4 * n)	TRCSSPCICR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCSTALLCTLR, Stall Control Register

The TRCSTALLCTLR characteristics are:

Purpose

Enables trace unit functionality that prevents trace unit buffer overflows.

Configuration

External register TRCSTALLCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCSTALLCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR3.STALLCTL == 1. Otherwise, direct accesses to TRCSTALLCTLR are RES0.

Attributes

TRCSTALLCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0														NOOVERFLOW					RES0			ISTALL		RES0			LEVEL				

Bits [31:14]

Reserved, RES0.

NOOVERFLOW, bit [13]

When TRCIDR3.NOOVERFLOW == 1:

Trace overflow prevention.

NOOVERFLOW	Meaning
0b0	Trace unit buffer overflow prevention is disabled.
0b1	Trace unit buffer overflow prevention is enabled.

Note that enabling this feature might cause a significant performance impact.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [12:9]

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCSTATR, Trace Status Register

The TRCSTATR characteristics are:

Purpose

Returns the trace unit status.

Configuration

External register TRCSTATR bits [31:0] are architecturally mapped to AArch64 System register [TRCSTATR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCSTATR are RES0.

Attributes

TRCSTATR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																PMSTABLE		IDLE													

Bits [31:2]

Reserved, RES0.

PMSTABLE, bit [1]

Programmers' model stable.

PMSTABLE	Meaning
0b0	The programmers' model is not stable.
0b1	The programmers' model is stable.

Accessing this field has the following behavior:

- When the trace unit is enabled, access to this field is **UNKNOWN/WI**.
- Otherwise, access to this field is **RO**.

IDLE, bit [0]

Idle status.

IDLE	Meaning
0b0	The trace unit is not idle.
0b1	The trace unit is idle.

(old)

htmldiff from-

(new)

TRCSYNCP, Synchronization Period Register

The TRCSYNCP characteristics are:

Purpose

Controls how often trace protocol synchronization requests occur.

Configuration

External register TRCSYNCP bits [31:0] are architecturally mapped to AArch64 System register [TRCSYNCP\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCSYNCP are RES0.

Attributes

TRCSYNCP is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																												PERIOD			

Bits [31:5]

Reserved, RES0.

PERIOD, bits [4:0]

Defines the number of bytes of trace between each periodic trace protocol synchronization request.

PERIOD	Meaning
0b00000	Trace protocol synchronization is disabled.
0b01000	Trace protocol synchronization request occurs after 2^8 bytes of trace.
0b01001	Trace protocol synchronization request occurs after 2^9 bytes of trace.
0b01010	Trace protocol synchronization request occurs after 2^{10} bytes of trace.
0b01011	Trace protocol synchronization request occurs after 2^{11} bytes of trace.
0b01100	Trace protocol synchronization request occurs after 2^{12} bytes of trace.
0b01101	Trace protocol synchronization request occurs after 2^{13} bytes of trace.
0b01110	Trace protocol synchronization request occurs after 2^{14} bytes of trace.
0b01111	Trace protocol synchronization request occurs after 2^{15} bytes of trace.
0b10000	Trace protocol synchronization request occurs after 2^{16} bytes of trace.
0b10001	Trace protocol synchronization request occurs after 2^{17} bytes of trace.
0b10010	Trace protocol synchronization request occurs after 2^{18} bytes of trace.
0b10011	Trace protocol synchronization request occurs after 2^{19} bytes of trace.
0b10100	Trace protocol synchronization request occurs after 2^{20} bytes of trace.

Other values are reserved. If a reserved value is programmed into PERIOD, then the behavior of the synchronization period counter is CONSTRAINED UNPREDICTABLE and one of the following behaviors occurs:

- No trace protocol synchronization requests are generated by this counter.
- Trace protocol synchronization requests occur at the specified period.
- Trace protocol synchronization requests occur at some other UNKNOWN period which can vary.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCSYNCP

Must be programmed if [TRCIDR3.SYNCP](#) == 0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCSYNCP can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x034	TRCSYNCP

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

(old)**htmldiff from-****(new)**

(old)

html diff from -

(new)

TRCTRAIDR, Trace ID Register

The TRCTRAIDR characteristics are:

Purpose

Sets the trace ID for instruction trace.

Configuration

External register TRCTRAIDR bits [31:0] are architecturally mapped to AArch64 System register [TRCTRAIDR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented and FEAT_TRC_EXT is** implemented. Otherwise, direct accesses to TRCTRAIDR are RES0.

Attributes

TRCTRAIDR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0														TRACEID																	

Bits [31:7]

Reserved, RES0.

TRACEID, bits [6:0]

Trace ID field. Sets the trace ID value for instruction trace. The width of the field is indicated by the value of [TRCIDR5](#).TRACEIDSIZE. Unimplemented bits are RES0.

If an implementation supports AMBA ATB, then:

- The width of the field is 7 bits.
- Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure.

See the AMBA ATB Protocol Specification for information about which ATID values are reserved.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCTRAIDR

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCTRACEIDR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x040	TRCTRACEIDR

This interface is accessible as follows:

- When OSLockStatus(), or !IsTraceCorePowered() or !AllowExternalTraceAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

TRCTSCTLR, Timestamp Control Register

The TRCTSCTLR characteristics are:

Purpose

Controls the insertion of global timestamps in the trace stream.

Configuration

External register TRCTSCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCTSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and TRCIDR0.TSSIZE != 0b00000. Otherwise, direct accesses to TRCTSCTLR are RES0.

Attributes

TRCTSCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								EVENT_TYPE		RES0		EVENT_SEL			

Bits [31:8]

Reserved, RES0.

EVENT_TYPE, bit [7] When TRCIDR4.NUMRSPAIR != 0b0000:

Chooses the type of Resource Selector.

EVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCTSCTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCTSCTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCTSCTLR.EVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCVICTLR, ViewInst Main Control Register

The TRCVICTLR characteristics are:

Purpose

Controls instruction trace filtering.

Configuration

External register TRCVICTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCVICTLR\[31:0\]](#).

This register is present only when FEAT_ETE is **implemented** and FEAT_TRC_EXT is **implemented**. Otherwise, direct accesses to TRCVICTLR are RES0.

Attributes

TRCVICTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20
RES0	EXLEVEL_RL_EL2	EXLEVEL_RL_EL1	EXLEVEL_RL_EL0	RES0	EXLEVEL_NS_EL2	EXLEVEL_NS_EL1	EXLEVEL_NS_EL0	RES0	EXLEVEL_RL_EL2	EXLEVEL_RL_EL1	EXLEVEL_RL_EL0

Bits [31:27]

Reserved, RES0.

EXLEVEL_RL_EL2, bit [26]

When TRCIDR6.EXLEVEL_RL_EL2 == 1:

Filter instruction trace for EL2 in Realm state.

EXLEVEL_RL_EL2	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL2 is 0 the trace unit generates instruction trace for EL2 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL2 is 1 the trace unit does not generate instruction trace for EL2 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL2 is 0 the trace unit does not generate instruction trace for EL2 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL2 is 1 the trace unit generates instruction trace for EL2 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL1, bit [25]**When TRCIDR6.EXLEVEL_RL_EL1 == 1:**

Filter instruction trace for EL1 in Realm state.

EXLEVEL_RL_EL1	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL1 is 0 the trace unit generates instruction trace for EL1 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL1 is 1 the trace unit does not generate instruction trace for EL1 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL1 is 0 the trace unit does not generate instruction trace for EL1 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL1 is 1 the trace unit generates instruction trace for EL1 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_RL_EL0, bit [24]**When TRCIDR6.EXLEVEL_RL_EL0 == 1:**

Filter instruction trace for EL0 in Realm state.

EXLEVEL_RL_EL0	Meaning
0b0	When TRCVICTLR.EXLEVEL_NS_EL0 is 0 the trace unit generates instruction trace for EL0 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL0 is 1 the trace unit does not generate instruction trace for EL0 in Realm state.
0b1	When TRCVICTLR.EXLEVEL_NS_EL0 is 0 the trace unit does not generate instruction trace for EL0 in Realm state. When TRCVICTLR.EXLEVEL_NS_EL0 is 1 the trace unit generates instruction trace for EL0 in Realm state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [23]

Reserved, RES0.

EXLEVEL_NS_EL2, bit [22]**When Non-secure EL2 is implemented:**

Filter instruction trace for EL2 in Non-secure state.

EXLEVEL_NS_EL2	Meaning
0b0	The trace unit generates instruction trace for EL2 in Non-secure state.
0b1	The trace unit does not generate instruction trace for EL2 in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_EL1, bit [21]**When Non-secure EL1 is implemented:**

Filter instruction trace for EL1 in Non-secure state.

EXLEVEL_NS_EL1	Meaning
0b0	The trace unit generates instruction trace for EL1 in Non-secure state.
0b1	The trace unit does not generate instruction trace for EL1 in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_NS_ELO, bit [20]**When Non-secure ELO is implemented:**

Filter instruction trace for ELO in Non-secure state.

EXLEVEL_NS_ELO	Meaning
0b0	The trace unit generates instruction trace for ELO in Non-secure state.
0b1	The trace unit does not generate instruction trace for ELO in Non-secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL3, bit [19]**When EL3 is implemented:**

Filter instruction trace for EL3.

EXLEVEL_S_EL3	Meaning
0b0	The trace unit generates instruction trace for EL3.
0b1	The trace unit does not generate instruction trace for EL3.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL2, bit [18]**When EL2 is implemented and FEAT_SEL2 is implemented:**

Filter instruction trace for EL2 in Secure state.

EXLEVEL_S_EL2	Meaning
0b0	The trace unit generates instruction trace for EL2 in Secure state.
0b1	The trace unit does not generate instruction trace for EL2 in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_EL1, bit [17]**When Secure EL1 is implemented:**

Filter instruction trace for EL1 in Secure state.

EXLEVEL_S_EL1	Meaning
0b0	The trace unit generates instruction trace for EL1 in Secure state.
0b1	The trace unit does not generate instruction trace for EL1 in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

EXLEVEL_S_ELO, bit [16]**When Secure ELO is implemented:**

Filter instruction trace for ELO in Secure state.

EXLEVEL_S_ELO	Meaning
0b0	The trace unit generates instruction trace for ELO in Secure state.
0b1	The trace unit does not generate instruction trace for ELO in Secure state.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [15:12]

Reserved, RES0.

TRCERR, bit [11]**When TRCIDR3.TRCERR == 1:**

Controls the forced tracing of System Error exceptions.

TRCERR	Meaning
0b0	Forced tracing of System Error exceptions is disabled.
0b1	Forced tracing of System Error exceptions is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TRCRESET, bit [10]

Controls the forced tracing of PE Resets.

TRCRESET	Meaning
0b0	Forced tracing of PE Resets is disabled.
0b1	Forced tracing of PE Resets is enabled.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

SSSTATUS, bit [9]

ViewInst start/stop function status.

SSSTATUS	Meaning
0b0	Stopped State. The ViewInst start/stop function is in the stopped state.
0b1	Started State. The ViewInst start/stop function is in the started state.

Before software enables the trace unit, it must write to this field to set the initial state of the ViewInst start/stop function. If the ViewInst start/stop function is not used then set this field to 1. Arm recommends that the value of this field is set before each trace session begins.

If the trace unit becomes disabled while a start point or stop point is still speculative, then the value of TRCVICTLR.SSSTATUS is UNKNOWN and might represent the result of a speculative start point or stop point.

If software which is running on the PE being traced disables the trace unit, either by clearing [TRCPRGCTLR.EN](#) or locking the OS Lock, Arm recommends that a DSB and an ISB instruction are executed before disabling the trace unit to prevent any start points or stop points being speculative at the point of disabling the trace unit. This procedure assumes that all start points or stop points occur before the barrier instructions are executed. The procedure does not guarantee that there are no speculative start points or stop points when disabling, although it helps minimize the probability.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- Access is **RES1** if all of the following are true:
 - TRCIDR4.NUMACPAIRS == 0b0000
 - TRCIDR4.NUMPC == 0b0000
- Otherwise, access to this field is **RW**.

Bit [8]

Reserved, RES0.

EVENT_TYPE, bit [7]

When TRCIDR4.NUMRSPAIR != 0b0000:

Chooses the type of Resource Selector.

EVENT_TYPE	Meaning
0b0	A single Resource Selector. TRCVICTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event.
0b1	A Boolean-combined pair of Resource Selectors. TRCVICTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCVICTLR.EVENT.SEL[4] is RES0.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [6:5]

Reserved, RES0.

(old)

htmldiff from-

(new)

TRCVIIECTLR, ViewInst Include/Exclude Control Register

The TRCVIIECTLR characteristics are:

Purpose

Use this to select, or read, the Address Range Comparators for the ViewInst include/exclude function.

Configuration

External register TRCVIIECTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCVIIECTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR4.NUMACPAIRS) > 0$. Otherwise, direct accesses to TRCVIIECTLR are RES0.

Attributes

TRCVIIECTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								EXCLUDE[7]		EXCLUDE[6]		EXCLUDE[5]		EXCLUDE[4]		EXCLUDE[3]		EXCLUDE[2]		EXCLUDE[1]		EXCLUDE[0]									

Bits [31:24]

Reserved, RES0.

EXCLUDE[<m>], bit [m+16], for m = 7 to 0

Exclude Address Range Comparator <m>. Selects whether Address Range Comparator <m> is in use with the ViewInst exclude function.

EXCLUDE[<m>]	Meaning
0b0	The address range that Address Range Comparator <m> defines, is not selected for the ViewInst exclude function.
0b1	The address range that Address Range Comparator <m> defines, is selected for the ViewInst exclude function.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMACPAIRS}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Bits [15:8]

Reserved, RES0.

The TRCVIPCSSCTLR characteristics are:

Use this to select, or read, which PE Comparator Inputs can control the ViewInst start/stop function.

External register TRCVIPCSSCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCVIPCSSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and UInt(TRCIDR4.NUMPC) > 0. Otherwise, direct accesses to TRCVIPCSSCTLR are RES0.

TRCVIPCSSCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
RES0								STOP[7]	STOP[6]	STOP[5]	STOP[4]	STOP[3]	STOP[2]	STOP[1]	STOP[0]	RES0								START[7]	START[6]

Reserved, RES0.

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of stopping trace.

STOP[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a stop resource.
0b1	The PE Comparator Input <m>, is selected as a stop resource.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMPC}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Reserved, RES0.

START[<m>], bit [m], for m = 7 to 0

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of starting trace.

START[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a start resource.
0b1	The PE Comparator Input <m>, is selected as a start resource.

This bit is RES0 if $m \geq \text{TRCIDR4.NUMPC}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVIPCSSCTLR

Must be programmed if [TRCIDR4](#).NUMPC != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCVIPCSSCTLR can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x08C	TRCVIPCSSCTL

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVISSCTLR, ViewInst Start/Stop Control Register

The TRCVISSCTLR characteristics are:

Purpose

Use this to select, or read, the Single Address Comparators for the ViewInst start/stop function.

Configuration

External register TRCVISSCTLR bits [31:0] are architecturally mapped to AArch64 System register [TRCVISSCTLR\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and `UInt(TRCIDR4.NUMACPAIRS) > 0`. Otherwise, direct accesses to TRCVISSCTLR are RES0.

Attributes

TRCVISSCTLR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19
STOP[15]	STOP[14]	STOP[13]	STOP[12]	STOP[11]	STOP[10]	STOP[9]	STOP[8]	STOP[7]	STOP[6]	STOP[5]	STOP[4]	STOP[3]

STOP[<m>], bit [m+16], for m = 15 to 0

Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of stopping trace.

STOP[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected as a stop resource.
0b1	The Single Address Comparator <m>, is selected as a stop resource.

This bit is RES0 if $m \geq 2 \times$ [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

START[<m>], bit [m], for m = 15 to 0

Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of starting trace.

START[<m>]	Meaning
0b0	The Single Address Comparator <m>, is not selected as a start resource.
0b1	The Single Address Comparator <m>, is selected as a start resource.

This bit is RES0 if $m \geq 2 \times$ [TRCIDR4](#).NUMACPAIRS.

The reset behavior of this field is:

Must be programmed if [TRCIDR4](#).NUMACPAIRS > 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance
ETE	0x088	TRCVISSCTLR

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

TRCVMIDCCTLRO, Virtual Context Identifier Comparator Control Register 0

The TRCVMIDCCTLRO characteristics are:

Purpose

Virtual Context Identifier Comparator mask values for the [TRCVMIDCVR<n>](#) registers, where n=0-3.

Configuration

External register TRCVMIDCCTLRO bits [31:0] are architecturally mapped to AArch64 System register [TRCVMIDCCTLRO\[31:0\]](#).

This register is present only when FEAT ETE is implemented, **FEAT TRC_EXT is implemented**, $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 0x0$ and $\text{UInt}(\text{TRCIDR2.VMIDSIZE}) > 0$. Otherwise, direct accesses to TRCVMIDCCTLRO are RES0.

Attributes

TRCVMIDCCTLRO is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	
COMP3[7]	COMP3[6]	COMP3[5]	COMP3[4]	COMP3[3]	COMP3[2]	COMP3[1]	COMP3[0]	COMP2[7]	COMP2[6]	COMP2[5]	COMP2[4]

COMP3[<m>], bit [m+24], for m = 7 to 0
When $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 3$:

TRCVMIDCVR3 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR3. Each bit in this field corresponds to a byte in TRCVMIDCVR3.

COMP3[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR3[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR3[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP2[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 2:

TRCVMIDCVR2 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR2. Each bit in this field corresponds to a byte in TRCVMIDCVR2.

COMP2[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR2[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR2[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP1[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 1:

TRCVMIDCVR1 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR1. Each bit in this field corresponds to a byte in TRCVMIDCVR1.

COMP1[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR1[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR1[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP0[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 0:

TRCVMIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0.

COMP0[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.VMIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCVMIDCTRL0

If software uses the [TRCVMIDCVR<n>](#) registers, where n=0-3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCVMIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCVMIDCVR<n>](#) is not 0x00, the behavior of the Virtual Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCVMIDCCTRL0 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x688	TRCVMIDCCTLR0

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The TRCVMIDCCTL1 characteristics are:

Purpose

Virtual Context Identifier Comparator mask values for the [TRCVMIDCVR<n>](#) registers, where n=4-7.

Configuration

External register TRCVMIDCCTL1 bits [31:0] are architecturally mapped to AArch64 System register [TRCVMIDCCTL1\[31:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented**, $\text{UInt}(\text{TRCIDR4.NUMVMIDC}) > 0x4$ and $\text{UInt}(\text{TRCIDR2.VMIDSIZE}) > 0$. Otherwise, direct accesses to TRCVMIDCCTLR1 are RES0.

Attributes

TRCVMIDCCTL1 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21
COMP7[7]	COMP7[6]	COMP7[5]	COMP7[4]	COMP7[3]	COMP7[2]	COMP7[1]	COMP7[0]	COMP6[7]	COMP6[6]	COMP6[5]

COMP7[<m>], bit [m+24], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 7:

TRCVMIDCVR7 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR7. Each bit in this field corresponds to a byte in TRCVMIDCVR7.

COMP7[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR7[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR7[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.VMIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP6[<m>], bit [m+16], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 6:

TRCVMIDCVR6 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR6. Each bit in this field corresponds to a byte in TRCVMIDCVR6.

COMP6[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR6[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR6[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP5[<m>], bit [m+8], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 5:

TRCVMIDCVR5 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR5. Each bit in this field corresponds to a byte in TRCVMIDCVR5.

COMP5[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR5[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR5[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if m >= [TRCIDR2.VMIDSIZE](#).

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

COMP4[<m>], bit [m], for m = 7 to 0
When UInt(TRCIDR4.NUMVMIDC) > 4:

TRCVMIDCVR4 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR4. Each bit in this field corresponds to a byte in TRCVMIDCVR4.

COMP4[<m>]	Meaning
0b0	The trace unit includes TRCVMIDCVR4[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.
0b1	The trace unit ignores TRCVMIDCVR4[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.

This bit is RES0 if $m \geq \text{TRCIDR2.VMIDSIZE}$.

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Accessing TRCVMIDCTRL1

If software uses the [TRCVMIDCVR<n>](#) registers, where n=4-7, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in [TRCVMIDCVR<n>](#) to 0x00.

If any bit is 1 and the relevant byte in [TRCVMIDCVR<n>](#) is not 0x00, the behavior of the Virtual Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

TRCVMIDCTLR1 can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x68C	TRCVMIDCCTLR1

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n>, n = 0 - 7

The TRCVMIDCVR<n> characteristics are:

Purpose

Contains the Virtual Context Identifier Comparator value.

Configuration

External register TRCVMIDCVR<n> bits [63:0] are architecturally mapped to AArch64 System register [TRCVMIDCVR<n>\[63:0\]](#).

This register is present only when FEAT_ETE is implemented, **FEAT_TRC_EXT is implemented** and $UInt(TRCIDR4.NUMVMIDC) > n$. Otherwise, direct accesses to TRCVMIDCVR<n> are RES0.

Attributes

TRCVMIDCVR<n> is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																VALUE															
																VALUE															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VALUE, bits [63:0]

Virtual context identifier value. The width of this field is indicated by [TRCIDR2.VMIDSIZE](#). Unimplemented bits are RES0. After a PE Reset, the trace unit assumes that the Virtual context identifier is zero until the PE updates the Virtual context identifier .

The reset behavior of this field is:

- On a Trace unit reset, this field resets to an architecturally UNKNOWN value.

Accessing TRCVMIDCVR<n>

Must be programmed if any of the following are true:

- [TRCRSCTLR<a>](#).GROUP == 0b0111 and [TRCRSCTLR<a>](#).VMID[n] == 1.
- [TRCACATR<a>](#).CONTEXTTYPE == 0b10 or 0b11 and [TRCACATR<a>](#).CONTEXT == n.

TRCVMIDCVR<n> can be accessed through the external debug interface:

Component	Offset	Instance
ETE	0x640 + (8 * n)	TRCVMIDCVR<n>

This interface is accessible as follows:

- When OSLockStatus(), or !AllowExternalTraceAccess() or !IsTraceCorePowered(), accesses to this register generate an error response.

- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5808; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMU

The PMU characteristics are:

Configuration

[PMCCFILTR_ELO\[31:0\]](#) and [PMCCFILTR_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, [PMCCFILTR_ELO\[63:32\]](#) and [PMCCFILTR_ELO\[63:32\]](#) are mapped to each other.

[PMCCFILTR_ELO\[31:0\]](#) and [PMCCFILTR\[31:0\]](#) are mapped to each other.

[PMCCNTR_ELO\[63:0\]](#) and [PMCCNTR_ELO\[63:0\]](#) are mapped to each other.

[PMCCNTR_ELO\[63:0\]](#) and [PMCCNTR\[63:0\]](#) are mapped to each other.

When FEAT_PMUv3_SS is implemented, [PMCCNTSVR_EL1\[63:0\]](#) and [PMCCNTSVR_EL1\[63:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID0\[31:0\]](#) and [PMCEID0_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID0\[31:0\]](#) and [PMCEID0\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID1\[31:0\]](#) and [PMCEID1_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID1\[31:0\]](#) and [PMCEID1\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID2\[31:0\]](#) and [PMCEID0_ELO\[63:32\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID2\[31:0\]](#) and [PMCEID2\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT32 is implemented, [PMCEID3\[31:0\]](#) and [PMCEID3\[31:0\]](#) are mapped to each other.

[PMCNTEN\[31:0\]](#) and [PMCNTENSET_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTEN\[63:32\]](#) and [PMCNTENSET_ELO\[63:32\]](#) are mapped to each other.

[PMCNTEN\[31:0\]](#) and [PMCNTENSET\[31:0\]](#) are mapped to each other.

[PMCNTEN\[31:0\]](#) and [PMCNTENCLR_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTEN\[63:32\]](#) and [PMCNTENCLR_ELO\[63:32\]](#) are mapped to each other.

[PMCNTEN\[31:0\]](#) and [PMCNTENCLR\[31:0\]](#) are mapped to each other.

[PMCNTENCLR_ELO\[31:0\]](#) and [PMCNTENCLR_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTENCLR_ELO\[63:32\]](#) and [PMCNTENCLR_ELO\[63:32\]](#) are mapped to each other.

[PMCNTENCLR_ELO\[31:0\]](#) and [PMCNTENCLR\[31:0\]](#) are mapped to each other.

[PMCNTENCLR_ELO\[31:0\]](#) and [PMCNTENSET_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTENCLR_ELO\[63:32\]](#) and [PMCNTENSET_ELO\[63:32\]](#) are mapped to each other.

[PMCNTENCLR_ELO\[31:0\]](#) and [PMCNTENSET\[31:0\]](#) are mapped to each other.

[PMCNTENSET_ELO\[31:0\]](#) and [PMCNTENSET_ELO\[31:0\]](#) are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTENSET_EL0](#)[63:32] and [PMCNTENSET_EL0](#)[63:32] are mapped to each other.

[PMCNTENSET_EL0](#)[31:0] and [PMCNTENCLR_EL0](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMCNTENSET_EL0](#)[63:32] and [PMCNTENCLR_EL0](#)[63:32] are mapped to each other.

[PMCNTENSET_EL0](#)[31:0] and [PMCNTENCLR](#)[31:0] are mapped to each other.

[PMCNTENSET_EL0](#)[31:0] and [PMCNTENSET](#)[31:0] are mapped to each other.

[PMCR_EL0](#)[7:0] and [PMCR_EL0](#)[7:0] are mapped to each other.

[PMCR_EL0](#)[7:0] and [PMCR](#)[7:0] are mapped to each other.

[PMEVCNTR<n>_EL0](#)[31:0] and [PMEVCNTR<n>_EL0](#)[31:0] are mapped to each other.

When FEAT_PMUv3p5 is implemented, [PMEVCNTR<n>_EL0](#)[63:32] and [PMEVCNTR<n>_EL0](#)[63:32] are mapped to each other.

[PMEVCNTR<n>_EL0](#)[31:0] and [PMEVCNTR<n>](#)[31:0] are mapped to each other.

When FEAT_PMUv3_SS is implemented, [PMEVCNTSVR<n>_EL1](#)[63:0] and [PMEVCNTSVR<n>_EL1](#)[63:0] are mapped to each other.

[PMEVTPER<n>_EL0](#)[63:0] and [PMEVTPER<n>_EL0](#)[63:0] are mapped to each other.

[PMEVTPER<n>_EL0](#)[31:0] and [PMEVTPER<n>](#)[31:0] are mapped to each other.

When FEAT_PMUv3_ICNTR is implemented, [PMICFILTR_EL0](#)[63:0] and [PMICFILTR_EL0](#)[63:0] are mapped to each other.

When FEAT_PMUv3_ICNTR is implemented and FEAT_PMUv3_SS is implemented, [PMICNTSVR_EL1](#)[63:0] and [PMICNTSVR_EL1](#)[63:0] are mapped to each other.

When FEAT_PMUv3_ICNTR is implemented, [PMICNTR_EL0](#)[63:0] and [PMICNTR_EL0](#)[63:0] are mapped to each other.

[PMINTEN](#)[31:0] and [PMINTENSET_EL1](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMINTEN](#)[63:32] and [PMINTENSET_EL1](#)[63:32] are mapped to each other.

[PMINTEN](#)[31:0] and [PMINTENSET](#)[31:0] are mapped to each other.

[PMINTEN](#)[31:0] and [PMINTENCLR_EL1](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMINTEN](#)[63:32] and [PMINTENCLR_EL1](#)[63:32] are mapped to each other.

[PMINTEN](#)[31:0] and [PMINTENCLR](#)[31:0] are mapped to each other.

[PMINTENCLR_EL1](#)[31:0] and [PMINTENCLR_EL1](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMINTENCLR_EL1](#)[63:32] and [PMINTENCLR_EL1](#)[63:32] are mapped to each other.

[PMINTENCLR_EL1](#)[31:0] and [PMINTENCLR](#)[31:0] are mapped to each other.

[PMINTENCLR_EL1](#)[31:0] and [PMINTENSET_EL1](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMINTENCLR_EL1](#)[63:32] and [PMINTENSET_EL1](#)[63:32] are mapped to each other.

[PMINTENCLR_EL1](#)[31:0] and [PMINTENSET](#)[31:0] are mapped to each other.

[PMINTENSET_EL1](#)[31:0] and [PMINTENSET_EL1](#)[31:0] are mapped to each other.

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented, [PMINTENSET_EL1](#)[63:32] and [PMINTENSET_EL1](#)[63:32] are mapped to each other.

[PMINTENSET_EL1](#)[31:0] and [PMINTENSET](#)[31:0] are mapped to each other.

[PMINTENSET_EL1](#)[31:0] and [PMINTENCLR_EL1](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMINTENSET_EL1](#)[63:32] and [PMINTENCLR_EL1](#)[63:32] are mapped to each other.

[PMINTENSET_EL1](#)[31:0] and [PMINTENCLR](#)[31:0] are mapped to each other.

[PMOVS](#)[31:0] and [PMOVSSET_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVS](#)[63:32] and [PMOVSSET_EL0](#)[63:32] are mapped to each other.

[PMOVS](#)[31:0] and [PMOVSSET](#)[31:0] are mapped to each other.

[PMOVS](#)[31:0] and [PMOVSLR_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVS](#)[63:32] and [PMOVSLR_EL0](#)[63:32] are mapped to each other.

[PMOVS](#)[31:0] and [PMOVSR](#)[31:0] are mapped to each other.

[PMOVSLR_EL0](#)[31:0] and [PMOVSLR_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVSLR_EL0](#)[63:32] and [PMOVSLR_EL0](#)[63:32] are mapped to each other.

[PMOVSLR_EL0](#)[31:0] and [PMOVSR](#)[31:0] are mapped to each other.

[PMOVSLR_EL0](#)[31:0] and [PMOVSSET_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVSLR_EL0](#)[63:32] and [PMOVSSET_EL0](#)[63:32] are mapped to each other.

[PMOVSLR_EL0](#)[31:0] and [PMOVSSET](#)[31:0] are mapped to each other.

[PMOVSSET_EL0](#)[31:0] and [PMOVSSET_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVSSET_EL0](#)[63:32] and [PMOVSSET_EL0](#)[63:32] are mapped to each other.

[PMOVSSET_EL0](#)[31:0] and [PMOVSSET](#)[31:0] are mapped to each other.

[PMOVSSET_EL0](#)[31:0] and [PMOVSLR_EL0](#)[31:0] are mapped to each other.

When FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented, [PMOVSSET_EL0](#)[63:32] and [PMOVSLR_EL0](#)[63:32] are mapped to each other.

[PMOVSSET_EL0](#)[31:0] and [PMOVSR](#)[31:0] are mapped to each other.

[PMSWINC_EL0](#)[31:0] and [PMSWINC_EL0](#)[31:0] are mapped to each other.

[PMSWINC_EL0](#)[31:0] and [PMSWINC](#)[31:0] are mapped to each other.

Attributes

PMU is a block of size: 4096 bytes

Contents

Offset	Name	Accessor condition	Register condition	M per a
0x000 + (8 * n) for n in 30:0	PMEVCNTR<n>_EL0	-	When FEAT_PMuV3_EXT is implemented	RW

PMU

0x0F8	PMCCNTR_EL0	-	When FEAT_PMUv3_EXT is implemented	RW
0x100	PMICNTR_EL0	When FEAT_PMUv3_ICNTR is implemented	When FEAT_PMUv3_ICNTR is implemented	RW
0x200	PMPCSR	-	When FEAT_PMUv3_EXT is implemented and FEAT_PCSRv8p2 is implemented	R0
0x208	PMCID1SR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and FEAT_PCSRv8p2 is implemented	R0
0x208	PMVCIDSR	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT64 is implemented and FEAT_PCSRv8p2 is implemented	R0
0x20C	PMVIDSR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented, FEAT_PCSRv8p2 is implemented and EL2 is implemented	R0
0x220	PMPCSR	-	When FEAT_PMUv3_EXT is implemented and FEAT_PCSRv8p2 is implemented	R0
0x228	PMCID1SR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and FEAT_PCSRv8p2 is implemented	R0
0x228	PMCCIDSR	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT is implemented	R0
0x22C	PMCID2SR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0x230	PMPCCTL	When FEAT_PCSRv8p9 is implemented	When FEAT_PCSRv8p9 is implemented	RW
0x400 + (4 * n) for n in 30:0	PMEVTYPER<n>_EL0[31:0]	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0x400 + (8 * n) for n in 30:0	PMEVTYPER<n>_EL0[63:0]	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0x47C	PMCCFILTR_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW

PMU

0x480	PMICFILTR_EL0	When FEAT_PMUv3_ICNTR is implemented	When FEAT_PMUv3_ICNTR is implemented	RW
0x4F8	PMCCFILTR_EL0	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0x600 + (8 * n) for n in 30:0	PMEVCNTSVR<n>_EL1	When FEAT_PMUv3_SS is implemented	When FEAT_PMUv3_SS is implemented	R0
0x6F8	PMCCNTSVR_EL1	When FEAT_PMUv3_SS is implemented	When FEAT_PMUv3_SS is implemented	R0
0x700	PMICNTSVR_EL1	When FEAT_PMUv3_SS is implemented and FEAT_PMUv3_ICNTR is implemented	When FEAT_PMUv3_ICNTR is implemented and FEAT_PMUv3_SS is implemented	R0
0xA00 + (4 * n) for n in 30:0	PMEVTYPEPER<n>_EL0[63:32]	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC00	PMCNTENSET_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC04	PMCNTENSET_EL0[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented)	When FEAT_PMUv3_EXT is implemented	RW
0xC10	PMCNTEN	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT64 is implemented	RW
0xC20	PMCNTENCLR_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC24	PMCNTENCLR_EL0[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented)	When FEAT_PMUv3_EXT is implemented	RW
0xC40	PMINTENSET_EL1	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC44	PMINTENSET_EL1[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or	When FEAT_PMUv3_EXT is implemented	RW

PMU

		FEAT_PMUv3_ICNTR is implemented)		
0xC50	PMINTEN	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT64 is implemented	RW
0xC60	PMINTENCLR_EL1	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC64	PMINTENCLR_EL1[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented)	When FEAT_PMUv3_EXT is implemented	RW
0xC80	PMOVSLR_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xC84	PMOVSLR_EL0[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented)	When FEAT_PMUv3_EXT is implemented	RW
0xC90	PMOVS	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT64 is implemented	RW
0xCA0	PMSWINC_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and an implementation implements PMSWINC_EL0	W0
0xCC0	PMOVSSET_EL0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xCC4	PMOVSSET_EL0[63:32]	When FEAT_PMUv3_EXT32 is implemented and (FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented)	When FEAT_PMUv3_EXT is implemented	RW
0xCE0	PMCGCR0	When FEAT_PMUv3_ICNTR is implemented	When FEAT_PMUv3_ICNTR is implemented	R0
0xE00	PMCFGR	-	When FEAT_PMUv3_EXT is implemented	R0

PMU

0xE04	PMCR_ELO	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xE08	PMIIDR	When FEAT_PMUv3_EXT64 is implemented	When (FEAT_PMUv3_EXT32 is implemented and an implementation implements PMIIDR) or FEAT_PMUv3_EXT64 is implemented	R0
0xE10	PMCR_ELO	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT is implemented	RW
0xE20	PMCEID0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0xE24	PMCEID1	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0xE28	PMCEID2	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p1 is implemented	R0
0xE2C	PMCEID3	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p1 is implemented	R0
0xE30	PMSSCR_EL1	When FEAT_PMUv3_SS is implemented	When FEAT_PMUv3_SS is implemented	RW
0xE40	PMMIR	-	When FEAT_PMUv3_EXT is implemented and FEAT_PMUv3p4 is implemented	R0
0xF00	PMITCTRL	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented and an implementation implements PMITCTRL	RW
0xFA8	PMDEVAFF	When FEAT_PMUv3_EXT64 is implemented	When FEAT_PMUv3_EXT64 is implemented	R0
0xFA8	PMDEVAFF0	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0xFAC	PMDEVAFF1	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0

PMU

0xFB0	PMLAR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	W0
0xFB4	PMLSR	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0xFB8	PMAUTHSTATUS	-	When FEAT_PMUv3_EXT is implemented	R0
0xFBC	PMDEVARCH	-	When FEAT_PMUv3_EXT is implemented	R0
0xFC8	PMDEVID	When FEAT_PMUv3_EXT32 is implemented	When FEAT_PMUv3_EXT32 is implemented	R0
0xFCC	PMDEVTYPE	-	When an implementation implements PMDEVTYPE	R0
0xFD0	PMPIDR4	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR4	R0
0xFE0	PMPIDR0	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR0	R0
0xFE4	PMPIDR1	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR1	R0
0xFE8	PMPIDR2	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR2	R0
0xFEC	PMPIDR3	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR3	R0
0xFF0	PMCIDR0	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR0	R0
0xFF4	PMCIDR1	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR1	R0
0xFF8	PMCIDR2	-	When FEAT_PMUv3_EXT is implemented and an implementation	R0

PMU

			implements PMCIDR2	
0xFFC	PMCIDR3	-	When FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR3	R0

Direct accesses to other offsets in this block are RES0.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file	htmldiff from-	(new)
-------------	----------------	-------

(old)

htmldiff from-

(new)

PMAUTHSTATUS, Performance Monitors Authentication Status register

The PMAUTHSTATUS characteristics are:

Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for Performance Monitors.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMAUTHSTATUS are RES0.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is OPTIONAL, and is required for CoreSight compliance. Arm recommends that this register is implemented.

Attributes

PMAUTHSTATUS is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				RTNID		RTID		RES0								RLNID		RLID		RES0				SNID		SID		NSNID		NSID	

Bits [31:28]

Reserved, RES0.

RTNID, bits [27:26]

Root non-invasive debug.

This field has the same value as DBGAUTHSTATUS_EL1.RTNID.

RTID, bits [25:24]

Root invasive debug.

RTID	Meaning
0b00	Not implemented.

Bits [23:16]

Reserved, RES0.

RLNID, bits [15:14]

Realm non-invasive debug.

This field has the same value as [DBGAUTHSTATUS_EL1.RLNID](#).

RLID, bits [13:12]

Realm invasive debug.

RLID	Meaning
0b00	Not implemented.

Bits [11:8]

Reserved, RES0.

SNID, bits [7:6]

Holds the same value as [DBGAUTHSTATUS_EL1.SNID](#).

SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

SID	Meaning
0b00	Not implemented.

All other values are reserved.

NSNID, bits [3:2]

Holds the same value as [DBGAUTHSTATUS_EL1.NSNID](#).

NSID, bits [1:0]

Non-secure invasive debug. Possible values of this field are:

NSID	Meaning
0b00	Not implemented.

All other values are reserved.

Accessing PMAUTHSTATUS

PMAUTHSTATUS can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFB8	PMAUTHSTATUS

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMAUTHSTATUS

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFB8

PMAUTHSTATUS can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xFB8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 15:17:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCCFILTR_EL0, Performance Monitors Cycle Counter Filter Register

The PMCCFILTR_EL0 characteristics are:

Purpose

Determines the modes in which the Cycle Counter, `PMU.PMCCNTR_EL0`, increments. `PMCCNTR_EL0`, increments.

Configuration

External register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch64 System register `PMCCFILTR_EL0[31:0]`.

External register PMCCFILTR_EL0 bits [63:32] are architecturally mapped to AArch64 System register `PMCCFILTR_EL0[63:32]` when FEAT_PMUv3_EXT64 is implemented.

External register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to AArch32 System register `PMCCFILTR[31:0]`.

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCCFILTR_EL0 are RES0.

PMCCFILTR_EL0 is in the Core power domain.

On a Warm or Cold reset, RW fields in this register reset to:

- Architecturally UNKNOWN values if the reset is to an Exception level that is using AArch64.
- 0 if the reset is to an Exception level that is using AArch32.

The register is not affected by an External debug reset.

Attributes

PMCCFILTR_EL0 is a `32-bit register`.

- 64-bit register when FEAT_PMUv3_EXT64 is implemented
- 32-bit register otherwise

This register is part of the `PMU` block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
P	U	NSK	NSU	NSH	M	RES0	SH	T	RLK	RLU	RLH	RES0																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	U	NSK	NSU	NSH	M	RES0	SH	RES0	RLK	RLU	RLH	RES0																			

Bits [63:32]

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

P	Meaning
0b0	Count cycles in EL1.
0b1	Do not count cycles in EL1.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

U	Meaning
0b0	Count cycles in EL0.
0b1	Do not count cycles in EL0.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

NSH	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

Otherwise:

Reserved, RES0.

M, bit [26]**When EL3 is implemented:**

Secure EL3 filtering bit.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

Note

This field is not visible in the AArch32 [PMCCFILTR](#) System register.

Otherwise:

Reserved, RES0.

Bit [25]

Reserved, RES0.

SH, bit [24]**When FEAT_SEL2 is implemented and EL3 is implemented:**

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted.

Otherwise, cycles in Secure EL2 are not counted.

Note

This field is not visible in the AArch32 [PMCCFILTR](#) System register.

Otherwise:

Reserved, RES0.

T, bit [23]**When FEAT_TME is implemented:**

Transactional state filtering bit. Controls counting of Attributable events in Non-transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count Attributable events in Non-transactional state.

For each Unattributable event, it is IMPLEMENTATION DEFINED whether the filtering applies.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted.

Otherwise, cycles in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]

When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted.

Otherwise, cycles in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:0]

Reserved, RES0.

Accessing PMCCFILTR_EL0**Note**

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCCFILTR_EL0 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	U	NS	NS	UN	SH	M	RES0	SH	T	RLK	RLU	RLH															RES0				
Component						Offset						Instance																			
PMU						0x47C						PMCCFILTR_EL0																			

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMCCFILTR_EL0.RLK bit.

P	Meaning
0b0	Count cycles in EL1.
0b1	Do not count cycles in EL1.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMCCFILTR_EL0.RLU bit.

U	Meaning
0b0	Count cycles in EL0.
0b1	Do not count cycles in EL0.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMCCFILTR_EL0.RLH bit.

NSH	Meaning
0b0	Do not count cycles in EL2.
0b1	Count cycles in EL2.

Otherwise:

Reserved, RES0.

M, bit [26]**When EL3 is implemented:**

Secure EL3 filtering bit.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

Note

This field is not visible in the AArch32 [PMCCFILTR](#) System register.

Otherwise:

Reserved, RES0.

Bit [25]

Reserved, RES0.

SH, bit [24]**When FEAT_SEL2 is implemented and EL3 is implemented:**

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted.

Otherwise, cycles in Secure EL2 are not counted.

NoteThis field is not visible in the AArch32 [PMCCFILTR](#) System register.**Otherwise:**

Reserved, RES0.

T, bit [23]**When FEAT_TME is implemented:**

Transactional state filtering bit. Controls counting of Attributable events in Non-transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count Attributable events in Non-transactional state.

For each Unattributable event, it is IMPLEMENTATION DEFINED whether the filtering applies.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]**When FEAT_RME is implemented:**

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Realm EL1 are counted.

Otherwise, cycles in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]**When FEAT_RME is implemented:**

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Realm EL0 are counted.

Otherwise, cycles in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]**When FEAT_RME is implemented:**

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Realm EL2 are counted.

Otherwise, cycles in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:0]

Reserved, RES0.

Accessing PMCCFILTR_EL0**Note**

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0x47C

PMCCFILTR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x47C

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0x4F8

PMCCFILTR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x4F8

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMCCIDSR, CONTEXTIDR_ELx Sample Register

The PMCCIDSR characteristics are:

Purpose

Contains the sampled value of [CONTEXTIDR_EL1](#) and [CONTEXTIDR_EL2](#), captured on reading PMU.PMPCSR.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCCIDSR are RES0.

Note

If FEAT_PCSRv8p2 is not implemented, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID](#).PCSample.

Attributes

PMCCIDSR is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
CONTEXTIDR_EL2																															
CONTEXTIDR_EL1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CONTEXTIDR_EL2, bits [63:32]

Context ID. The value of [CONTEXTIDR_EL2](#) that is associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample is generated:

- If the PE is not executing at EL3, EL2 is using AArch64, and EL2 is enabled in the current Security state, then this field is set to the Context ID sampled from [CONTEXTIDR_EL2](#).
- Otherwise, this field is set to an UNKNOWN value.

Because the value written to this field is an indirect read of [CONTEXTIDR_EL2](#), it is CONSTRAINED UNPREDICTABLE whether this field is set to the original or new value if PMU.PMPCSR samples:

- An instruction that writes to [CONTEXTIDR_EL2](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

CONTEXTIDR_EL1, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample is generated:

- If EL1 is using AArch64, then the Context ID is sampled from [CONTEXTIDR_EL1](#).
- If EL1 is using AArch32, then the Context ID is sampled from [CONTEXTIDR](#).
- If EL3 is implemented and is using AArch32, then [CONTEXTIDR](#) is a banked register and this register samples the current banked copy of [CONTEXTIDR](#) for the Security state that is associated with the most recent PMU.PMPCSR sample.

Because the value written to this register is an indirect read of CONTEXTIDR, it is **CONSTRAINED UNPREDICTABLE** whether this register is set to the original or new value if PMU.PMPCSR samples:

- An instruction that writes to CONTEXTIDR.
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally **UNKNOWN** value.

Accessing PMCCIDSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register **UNKNOWN**, see 'Permitted behavior that might make the PC Sample-based profiling registers **UNKNOWN**'.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0x228

PMCCIDSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x228

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCCNTR_EL0, Performance Monitors Cycle Counter

The PMCCNTR_EL0 characteristics are:

Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. For more information, see 'Time as measured by the Performance Monitors cycle counter'.

[PMCCFILTER_EL0](#) [PMU.PMCCFILTER_EL0](#) determines the modes and states in which the PMCCNTR_EL0 can increment.

Configuration

External register PMCCNTR_EL0 bits [63:0] are architecturally mapped to AArch64 System register [PMCCNTR_EL0\[63:0\]](#).

External register PMCCNTR_EL0 bits [63:0] are architecturally mapped to AArch32 System register [PMCCNTR\[63:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCCNTR_EL0 are RES0.

PMCCNTR_EL0 is in the Core power domain.

Attributes

PMCCNTR_EL0 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CCNT, bits [63:0]

Cycle count. Depending on the values of [PMU.PMCR_EL0.{LC,D}](#), the cycle count increments in one of the following ways: [PMCR_EL0.{LC,D}](#), the cycle count increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMU.PMCR_EL0.C](#) sets this field to 0. [PMCR_EL0.C](#) sets this field to 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCCNTR_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCCNTR_EL0 can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x0F8	PMCCNTR_EL0	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
PMU	0x0FC	PMCCNTR_EL0	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCCNTR_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x0F8

PMCCNTR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x0F8

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMCCNTSVR_EL1, Performance Monitors Cycle Count Saved Value Register

The PMCCNTSVR_EL1 characteristics are:

Purpose

Captures the PMU Cycle counter, PMU.PMCCNTR_EL0.

Configuration

External register PMCCNTSVR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [PMCCNTSVR_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMCCNTSVR_EL1 are RES0.

Attributes

PMCCNTSVR_EL1 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CCNT, bits [63:0]

Sampled Cycle Count. The value of PMU.PMCCNTR_EL0 at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCCNTSVR_EL1

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x6F8

PMCCNTSVR_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x6F8

- When !AllowExternalPMSSAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F.

For more information about the Common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Note

This view of the register was previously called PMCEID0_EL0.

Configuration

External register PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID0_EL0\[31:0\]](#).

External register PMCEID0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID0\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMCEID0 are RES0.

PMCEID0 is in the Core power domain.

Attributes

PMCEID0 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event n.

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID0

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCEID0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xE20	PMCEID0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCEID0

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xE20

PMCEID0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE20

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

(old)**htmldiff from-****(new)**

(old)

htmldiff from-

(new)

PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

For more information about the Common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Note

This view of the register was previously called PMCEID1_EL0.

Configuration

External register PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID1_EL0\[31:0\]](#).

External register PMCEID1 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID1\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMCEID1 are RES0.

PMCEID1 is in the Core power domain.

Attributes

PMCEID1 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
ID31	ID30	ID29	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7

ID<n>, bit [n], for n = 31 to 0

ID[n] corresponds to Common event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID1

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCEID1 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xE24	PMCEID1

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCEID1

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xE24

PMCEID1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE24

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configuration

External register PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID0_EL0\[63:32\]](#).

External register PMCEID2 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID2\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p1 is implemented. Otherwise, direct accesses to PMCEID2 are RES0.

PMCEID2 is in the Core power domain.

~~This register is present only when FEAT_PMUv3p1 is implemented. Otherwise, direct accesses to PMCEID2 are RES0.~~

Attributes

PMCEID2 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
IDhi31	IDhi30	IDhi29	IDhi28	IDhi27	IDhi26	IDhi25	IDhi24	IDhi23	IDhi22	IDhi21	IDhi20	IDhi19	IDhi18	IDhi17	IDhi16	IDhi15

IDhi<n>, bit [n], for n = 31 to 0

IDhi[n] corresponds to Common event (0x4000 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

(old)

htmldiff from-

(new)

PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

Purpose

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

For more information about the Common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

Configuration

External register PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID1_ELO\[63:32\]](#).

External register PMCEID3 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID3\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p1 is implemented. Otherwise, direct accesses to PMCEID3 are RES0.

PMCEID3 is in the Core power domain.

This register is present only when FEAT_PMUv3p1 is implemented. Otherwise, direct accesses to PMCEID3 are RES0.

Attributes

PMCEID3 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
IDhi31	IDhi30	IDhi29	IDhi28	IDhi27	IDhi26	IDhi25	IDhi24	IDhi23	IDhi22	IDhi21	IDhi20	IDhi19	IDhi18	IDhi17	IDhi16	IDhi15

IDhi<n>, bit [n], for n = 31 to 0

IDhi[n] corresponds to Common event (0x4020 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Common event is not implemented, or not counted.
0b1	The Common event is implemented.

When the value of a bit in the field is 1, the corresponding Common event is implemented and counted.

Note

Arm recommends that if a Common event is never counted, the value of the corresponding bit is 0.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional Common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing PMCEID3**Note**

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCEID3 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xE2C	PMCEID3

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCEID3**Note**

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xE2C

PMCEID3 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE2C

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCFGR, Performance Monitors Configuration Register

The PMCFGR characteristics are:

Purpose

Contains PMU-specific configuration data.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCFGR are RES0.

PMCFGR is in the Core power domain.

Attributes

PMCFGR is a: 32-bit register.

- 64-bit register when FEAT_PMUv3_EXT64 is implemented
- 32-bit register otherwise

This register is part of the PMU block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
NCG				RES0				SS	FZ	RES0	UEN	WT	NA	EX	CCD	CC	SIZE				N										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NCG				RES0				FZ		RES0	UEN	WT	NA	EX	CCD	CC	SIZE				N										

This feature is not supported, so this field is RAZ.

Reads as 0b0000.

Access to this field is RO.

Bits [27:22]

BitsNCG, bits [63:31]:3228

Reserved, RES0.

NCG, bits [31:28]

When FEAT_PMUv3p9 is implemented:

Counter Groups. Defines the number of counter groups implemented, minus one.

Reads as 0b0001.

Access to this field is **RO**.

Otherwise:

Defines the number of counter groups implemented, minus one.

This field reads-as-zero.

Reads as 0b0000.

Access to this field is **RO**.

Bits [27:23]

Reserved, RES0.

SS, bit [22]

Snapshot supported.

SS	Meaning
0b0	Snapshot mechanism not supported. The locations 0x600-0x7FC and 0xE30-0xE3C are IMPLEMENTATION DEFINED.
0b1	Snapshot mechanism supported. PMSVR<n> and PMSSSCR are implemented.

FEAT_PMUv3_SS implements the functionality identified by the value 1.

If FEAT_PMUv3_SS is not implemented, a PMU might include an IMPLEMENTATION DEFINED snapshot mechanism, including one using the IMPLEMENTATION DEFINED registers 0x600-0x7FC and 0xE30-0xE3C.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

FZO, bit [21]

Freeze-on-overflow supported. Defined values are:

FZO	Meaning
0b0	Freeze-on-overflow mechanism is not supported. PMU.PMCR_EL0.FZO is PMCR_EL0.FZO is RES0.
0b1	Freeze-on-overflow mechanism is supported. PMU.PMCR_EL0.FZO is RW. PMCR_EL0.FZO is RW.

FEAT_PMUv3p7 implements the functionality added by the value 0b1.

From Armv8.7, if FEAT_PMUv3 is implemented, the only permitted value is 0b1.

Bit [20]

Reserved, RES0.

UEN, bit [19]

User-mode Enable Register supported. [PMUSERENR_EL0](#) is not visible in the external debug interface, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

WT, bit [18]

This feature is not supported, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

NA, bit [17]

This feature is not supported, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

EX, bit [16]

Export supported. Value is IMPLEMENTATION DEFINED.

EX	Meaning
0b0	PMCR_EL0PMU.PMCR_EL0.X.X is RES0.
0b1	PMCR_EL0PMU.PMCR_EL0.X.X is read/write.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

CCD, bit [15]

Cycle counter has prescale.

This is RES1 if AArch32 is supported, and RAZ otherwise.

CCD	Meaning
0b0	PMCR_EL0PMU.PMCR_EL0.D.D is RES0.
0b1	PMCR_EL0PMU.PMCR_EL0.D.D is read/write.

CC, bit [14]

Dedicated cycle counter (counter 31) supported.

Reads as 0b1.

Access to this field is **RO**.

SIZE, bits [13:8]

Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit.

From Armv8, the largest counter is 64-bits, so the value of this field is 0b111111.

This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses.

Reads as 0b111111.

Access to this field is **RO**.

N, bits [7:0]

Number of counters, minus implemented one, in addition to the cycle counter, PMCCNTR_EL0. The maximum number of event counters is 31.

N	Meaning
0x00	Only PMU.PMCCNTR_EL0 implemented.
0x01..0x20	PMU.PMCCNTR_EL0 implemented, PMCCNTR_EL0 Number plus this number of event counters implemented, are 1 to 33 implemented.

All other values are reserved.

The count includes the optional Instruction Counter, PMU.PMICNTR_EL0.

The count includes the Cycle Counter, PMU.PMCCNTR_EL0. For example, if N == 0x00, there is a single counter, PMU.PMCCNTR_EL0, and PMEVCNTR0_EL0 is not implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMCFGR

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCFGR can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
NCG				RES0				SSFZ				OZRES0				UEN				WTNA				EXCCDCC				SIZE				N			
Component												Offset												Instance											
PMU												0xE00												PMCFGR											

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

NCG, bits [31:28]

When FEAT_PMUv3p9 is implemented:

Counter Groups. Defines the number of counter groups implemented, minus one.

Reads as 0b0001.

Access to this field is **RO**.

Otherwise:

Defines the number of counter groups implemented, minus one.

This field reads-as-zero.

Reads as 0b0000.

Access to this field is **RO**.

Bits [27:23]

Reserved, RES0.

SS, bit [22]

Snapshot supported.

SS	Meaning
0b0	Snapshot mechanism not supported. The locations 0x600-0x7FC and 0xE30-0xE3C are IMPLEMENTATION DEFINED.
0b1	Snapshot mechanism supported. PMSVR<n> and PMSSSCR are implemented.

FEAT_PMUv3_SS implements the functionality identified by the value 1.

If FEAT_PMUv3_SS is not implemented, a PMU might include an IMPLEMENTATION DEFINED snapshot mechanism, including one using the IMPLEMENTATION DEFINED registers 0x600-0x7FC and 0xE30-0xE3C.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

FZO, bit [21]

Freeze-on-overflow supported. Defined values are:

FZO	Meaning
0b0	Freeze-on-overflow mechanism is not supported. PMU.PMCR_EL0.FZO is RES0.
0b1	Freeze-on-overflow mechanism is supported. PMU.PMCR_EL0.FZO is RW.

FEAT_PMUv3p7 implements the functionality added by the value 0b1.

From Armv8.7, if FEAT_PMUv3 is implemented, the only permitted value is 0b1.

Bit [20]

Reserved, RES0.

UEN, bit [19]

User-mode Enable Register supported. [PMUSERENR_EL0](#) is not visible in the external debug interface, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

WT, bit [18]

This feature is not supported, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

NA, bit [17]

This feature is not supported, so this bit is RAZ.

Reads as 0b0.

Access to this field is **RO**.

EX, bit [16]

Export supported. Value is IMPLEMENTATION DEFINED.

EX	Meaning
0b0	PMU.PMCR_EL0.X is RES0.
0b1	PMU.PMCR_EL0.X is read/write.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

CCD, bit [15]

Cycle counter has prescale.

This is RES1 if AArch32 is supported, and RAZ otherwise.

CCD	Meaning
0b0	PMU.PMCR_EL0.D is RES0.
0b1	PMU.PMCR_EL0.D is read/write.

CC, bit [14]

Dedicated cycle counter (counter 31) supported.

Reads as 0b1.

Access to this field is **RO**.

SIZE, bits [13:8]

Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit.

From Armv8, the largest counter is 64-bits, so the value of this field is 0b111111.

This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses.

Reads as 0b111111.

Access to this field is **RO**.

N, bits [7:0]

Number of counters, minus one.

N	Meaning
0x00	Only PMU.PMCCNTR_EL0 implemented.
0x01..0x20	Number of counters implemented, 1 to 33.

All other values are reserved.

The count includes the optional Instruction Counter, PMU.PMICNTR_EL0.

The count includes the Cycle Counter, PMU.PMCCNTR_EL0. For example, if N == 0x00, there is a single counter, PMU.PMCCNTR_EL0, and PMEVCNTR0_EL0 is not implemented.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

no old file

htmldiff from-

(new)

PMCGCR0, Counter Group Configuration Register 0

The PMCGCR0 characteristics are:

Purpose

Encodes the number of PMU.PMEVCNTR<n>_EL0 counters implemented.

Configuration

This register is present only when FEAT_PMUv3_ICNTR is implemented. Otherwise, direct accesses to PMCGCR0 are RES0.

PMCGCR0 is in the Core power domain.

Attributes

PMCGCR0 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																CG1NC								CG0NC							

Bits [31:16]

Reserved, RES0.

CG1NC, bits [15:8]

Number of counters in group 1, which comprises the instruction counter [PMICNTR_EL0](#).

Reads as 0x01.

Access to this field is **RO**.

CG0NC, bits [7:0]

Number of counters in group 0, which comprises the event counters [PMEVCNTR<n>_EL0](#) and the cycle counter [PMCCNTR_EL0](#).

This field reads as [PMCFGR.N](#).

Accessing PMCGCR0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xCE0

PMCGCR0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xCE0

- When DoubleLockStatus(), or !IsCorePowered(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCID1SR, CONTEXTIDR_EL1 Sample Register

The PMCID1SR characteristics are:

Purpose

Contains the sampled value of [CONTEXTIDR_EL1](#), captured on reading [PMU.PMPCSR\[31:0\]](#). [PMPCSR\[31:0\]](#).

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented and FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMCID1SR are RES0.

If FEAT_PMUv3_EXT64 is implemented, the same content is present in the same location, and can be accessed using [PMCCIDSR\[31:0\]](#) or [PMCVIDSR\[31:0\]](#).

PMCID1SR is in the Core power domain.

This register is present only when FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMCID1SR are RES0.

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID](#).PCSample.

Attributes

PMCID1SR is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTEXTIDR_EL1																															

CONTEXTIDR_EL1, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent [PMU.PMPCSR](#) sample. When the most recent [PMU.PMPCSR](#) sample is generated: [PMPCSR](#) sample. When the most recent [PMPCSR](#) sample is generated:

- If EL1 is using AArch64, then the Context ID is sampled from [CONTEXTIDR_EL1](#).
- If EL1 is using AArch32, then the Context ID is sampled from [CONTEXTIDR](#).
- If EL3 is implemented and is using AArch32, then [CONTEXTIDR](#) is a banked register and [this register](#) [PMCID1SR](#) samples the current banked copy of [CONTEXTIDR](#) for the Security state that is associated with the most recent [PMU.PMPCSR](#) sample. [PMPCSR](#) sample.

Because the value written to [this register](#) [PMCID1SR](#) is an indirect read of [CONTEXTIDR](#), it is CONSTRAINED UNPREDICTABLE whether [this](#) [PMCID1SR](#) register is set to the original or new value if [PMU.PMPCSR](#) samples: [PMPCSR](#) samples:

- An instruction that writes to [CONTEXTIDR](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCID1SR

IMPLEMENTATION-DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

PMCID1SR can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x208	PMCID1SR

This interface is accessible as follows:

- When `IsCorePowered()`, `!DoubleLockStatus()` and `!OSLockStatus()`, accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance
PMU	0x228	PMCID1SR

This interface is accessible as follows:

- When `IsCorePowered()`, `!DoubleLockStatus()` and `!OSLockStatus()`, accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCID1SR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x208

PMCID1SR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x208

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

BlockAccess at address 0x228

PMCID1SR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x228

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

(old)

htmldiff from-

(new)

PMCID2SR, CONTEXTIDR_EL2 Sample Register

The PMCID2SR characteristics are:

Purpose

Contains the sampled value of [CONTEXTIDR_EL2](#), captured on reading [PMU.PMPCSR\[31:0\]](#).~~PMPCSR[31:0]~~.

Configuration

This [PMCID2SR](#) register is present in only the when Core FEAT_PMUv3_EXT32 power is implemented. Otherwise, direct accesses to PMCID2SR are ~~domain~~. RES0.

If FEAT_PMUv3_EXT64 is implemented, the same content is present in the same location, and can be accessed using [PMCCIDSR\[63:32\]](#).

~~This register is present only when FEAT_PCSRv8p2 is implemented and EL2 is implemented. Otherwise, direct accesses to PMCID2SR are RES0.~~

[PMCIDR2SR](#) is in the Core power domain.

Note

If FEAT_PCSRv8p2 is not implemented, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID](#).PCSample.

Attributes

PMCID2SR is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTEXTIDR_EL2																															

CONTEXTIDR_EL2, bits [31:0]

Context ID. The value of [CONTEXTIDR_EL2](#) that is associated with the most recent [PMU.PMPCSR](#) sample. When the most recent [PMU.PMPCSR](#) sample is generated: ~~PMPCSR~~ sample. When the most recent ~~PMPCSR~~ sample is generated:

- If the PE is not executing at EL3, EL2 is using AArch64, and EL2 is enabled in the current Security state, then this field is set to the Context ID sampled from [CONTEXTIDR_EL2](#).
- Otherwise, this field is set to an UNKNOWN value.

Because the value written to this field [PMCID2SR](#) is an indirect read of [CONTEXTIDR_EL2](#), it is CONSTRAINED UNPREDICTABLE whether this [PMCID2SR](#) field is set to the original or new value if [PMU.PMPCSR](#) samples: ~~PMPCSR~~ samples:

- An instruction that writes to [CONTEXTIDR_EL2](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance
PMU	0x22C	PMCID2SR

- When `IsCorePowered()`, `!DoubleLockStatus()` and `!OSLockStatus()`, accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Frame	Offset
PMU	0x22C

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

PMCIDR0, Performance Monitors Component Identification Register 0

The PMCIDR0 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configuration

This Implementation register of is this present register only when FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR0. Otherwise, direct accesses to PMCIDR0 are RES0 OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMCIDR0 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_0							

Bits [31:8]

Reserved, RES0.

PRMBL_0, bits [7:0]

Preamble.

Reads as 0x0D.

Access to this field is **RO**.

Accessing PMCIDR0

PMCIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFF0	PMCIDR0

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCIDR0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFF0

PMCIDR0 can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xFF0

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCIDR1, Performance Monitors Component Identification Register 1

The PMCIDR1 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configuration

ThisImplementation registerof isthis presentregister only when FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR1. Otherwise, direct accesses to PMCIDR1 are RES0OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMCIDR1 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								CLASS			PRMBL_1				

Bits [31:8]

Reserved, RES0.

CLASS, bits [7:4]

Component class.

CLASS	Meaning
0b1001	CoreSight component.

Other values are defined by the CoreSight Architecture.

This field reads as 0x9.

PRMBL_1, bits [3:0]

Preamble. RAZ.

Reads as 0b0000.

Access to this field is **RO**.

(old)

htmldiff from-

(new)

PMCIDR2, Performance Monitors Component Identification Register 2

The PMCIDR2 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configuration

This Implementation register of is this present register only when FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR2. Otherwise, direct accesses to PMCIDR2 are RES0 OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMCIDR2 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
												RES0																PRMBL_2											

Bits [31:8]

Reserved, RES0.

PRMBL_2, bits [7:0]

Preamble.

Reads as 0x05.

Access to this field is RO.

Accessing PMCIDR2

PMCIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFF8	PMCIDR2

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCIDR2

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFF8

PMCIDR2 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFF8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffdbbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCIDR3, Performance Monitors Component Identification Register 3

The PMCIDR3 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

Configuration

This Implementation register of is this present register only when FEAT_PMUv3_EXT is implemented and an implementation implements PMCIDR3. Otherwise, direct accesses to PMCIDR3 are RES0 OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMCIDR3 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PRMBL_3							

Bits [31:8]

Reserved, RES0.

PRMBL_3, bits [7:0]

Preamble.

Reads as 0xB1.

Access to this field is RO.

Accessing PMCIDR3

PMCIDR3 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFFC	PMCIDR3

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMCIDR3

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFFC

PMCIDR3 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFFC

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMCNTEN, Performance Monitors Count Enable register

The PMCNTEN characteristics are:

Purpose

Enables the Cycle Count Register, PMU.PMCCNTR_EL0, and any implemented event counters [PMEVCNTR<n>](#).

Configuration

External register PMCNTEN bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENSET_EL0\[31:0\]](#).

External register PMCNTEN bits [63:32] are architecturally mapped to AArch64 System register [PMCNTENSET_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTEN bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENSET\[31:0\]](#).

External register PMCNTEN bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENCLR_EL0\[31:0\]](#).

External register PMCNTEN bits [63:32] are architecturally mapped to AArch64 System register [PMCNTENCLR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTEN bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENCLR\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT64 is implemented. Otherwise, direct accesses to PMCNTEN are RES0.

Attributes

PMCNTEN is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

F0, bit [32]

When FEAT_PMUv3_ICNTR is implemented:

PMU.PMICNTR_EL0 counter enable. Enables the instruction counter.

F0	Meaning
0b0	PMU.PMICNTR_EL0 disabled.
0b1	PMU.PMICNTR_EL0 enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

C, bit [31]

PMU.PMCCNTR_EL0 enable. Enables the cycle counter register. Possible values are:

C	Meaning
0b0	PMU.PMCCNTR_EL0 is disabled.
0b1	PMU.PMCCNTR_EL0 enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter enable for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	PMU.PMEVCNTR<n>_EL0 is disabled.
0b1	PMU.PMEVCNTR<n>_EL0 is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTEN

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xC10

PMCNTEN can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC10

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

(old)

htmldiff from-

(new)

PMCNTENCLR_EL0, Performance Monitors Count Enable Clear register

The PMCNTENCLR_EL0 characteristics are:

Purpose

Disables the Cycle Count Register, PMU.PMCCNTR_EL0, and any implemented event counters, and any implemented event counters. PMEVCNTR<n>_EL0, PMEVCNTR<n>, PMCCNTR_EL0. Reading this register shows which counters are enabled.

Configuration

External register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to AArch64 System register PMCNTENCLR_EL0[31:0].

External register PMCNTENCLR_EL0 bits [63:32] are architecturally mapped to AArch64 System register PMCNTENCLR_EL0[63:32] when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register PMCNTENCLR[31:0].

External register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to AArch64 System register PMCNTENSET_EL0[31:0].

External register PMCNTENCLR_EL0 bits [63:32] are architecturally mapped to AArch64 System register PMCNTENSET_EL0[63:32] when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTENCLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register PMCNTENSET[31:0].

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCNTENCLR_EL0 are RES0.

PMCNTENCLR_EL0 is in the Core power domain.

Attributes

PMCNTENCLR_EL0 is a: 32-bit register.

- 64-bit register when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the PMU block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented:

63	31	62	30	61	29	60	28	59	27	58	26	57	25	56	24	55	23	54	22	53	21	52	20	51	19	50	18	49	17	48	16	47	15	46	14	45	13	44	12	43	11	42	10	41	9	
RES0																																														
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9																								

Bits [63:33]

Reserved, RES0.

F0, bit [32]

When FEAT_PMUv3_ICNTR is implemented:

PMU.PMICNTR_EL0 disable. On writes, allows software to disable PMU.PMICNTR_EL0. On reads, returns the PMU.PMICNTR_EL0 enable status.

F0	Meaning
0b0	PMU.PMICNTR_EL0 disabled.
0b1	PMU.PMICNTR_EL0 enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

PMCCNTR_EL0 PMU.PMCCNTR_EL0 disable bit. Disables the cycle counter register. Possible values are:

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, disables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter disable bit for PMU.PMEVCNTR<n>_EL0. **PMEVCNTR<n>_EL0**.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI. **PMCFGR.N** is less than 31, bits [30:PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_ELO is disabled. When written, has no effect. PMEVCNTR<n>_ELO is disabled. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_ELO is enabled. When written, disables PMU.PMEVCNTR<n>_ELO. PMEVCNTR<n>_ELO is enabled. When written, disables PMEVCNTR<n>_ELO.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENCLR_ELO

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCNTENCLR_ELO can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xC20								PMCNTENCLR_ELO															

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

C, bit [31]

PMU.PMCCNTR_ELO disable bit. Disables the cycle counter register.

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, disables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter disable bit for PMU.PMEVCNTR<n>_ELO.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_ELO is disabled. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_ELO is enabled. When written, disables PMU.PMEVCNTR<n>_ELO.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENCLR_ELO

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xC20

PMCNTENCLR_ELO can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC20

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xC20

PMCNTENCLR_ELO can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC20	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xC24

PMCNTENCLR_ELO can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC24	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbd6d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCNTENSET_EL0, Performance Monitors Count Enable Set register

The PMCNTENSET_EL0 characteristics are:

Purpose

Enables the Cycle Count Register, PMU.PMCCNTR_EL0, and any implemented event counters PMU.PMEVCNTR<n>_EL0. Reading this register shows which counters are enabled. ~~PMCCNTR_EL0, and any implemented event counters PMEVCNTR<n>_EL0. Reading this register shows which counters are enabled.~~

Configuration

External register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENSET_EL0\[31:0\]](#).

External register PMCNTENSET_EL0 bits [63:32] are architecturally mapped to AArch64 System register [PMCNTENSET_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMCNTENCLR_EL0\[31:0\]](#).

External register PMCNTENSET_EL0 bits [63:32] are architecturally mapped to AArch64 System register [PMCNTENCLR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENCLR\[31:0\]](#)~~PMCNTENSET[31:0]~~.

External register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCNTENSET\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCNTENSET_EL0 are RES0.

PMCNTENSET_EL0 is in the Core power domain.

Attributes

PMCNTENSET_EL0 is a: ~~32-bit register.~~

- 64-bit register when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented:

63	31	62	30	61	29	60	28	59	27	58	26	57	25	56	24	55	23	54	22	53	21	52	20	51	19	50	18	49	17	48	16	47	15	46	14	45	13	44	12	43	11	42	10	41	9	
RES0																																														
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9																								

Bits [63:33]

Reserved, RES0.

F0, bit [32]

When FEAT_PMUv3_ICNTR is implemented:

PMU.PMICNTR_EL0 enable. On writes, allows software to enable PMU.PMICNTR_EL0. On reads, returns the PMU.PMICNTR_EL0 enable status.

F0	Meaning
0b0	PMU.PMICNTR_EL0 disabled.
0b1	PMU.PMICNTR_EL0 enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **W1S**.

Otherwise:

Reserved, RES0.

C, bit [31]

PMCCNTR_EL0 PMU.PMCCNTR_EL0 enable bit. Enables the cycle counter register. Possible values are:

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, enables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter enable bit for PMU.PMEVCNTR<n>_EL0. **PMEVCNTR<n>_EL0.**

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI. **PMCFGR.N is less than 31, bits [30:PMCFGR.N] are RAZ/WI.**

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 is disabled. When written, has no effect. PMEVCNTR<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 event counter is enabled. When written, enables PMU.PMEVCNTR<n>_EL0. PMEVCNTR<n>_EL0 event counter is enabled. When written, enables PMEVCNTR<n>_EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENSET_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCNTENSET_EL0 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xC00								PMCNTENSET_EL0															

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

C, bit [31]

PMU.PMCCNTR_EL0 enable bit. Enables the cycle counter register.

C	Meaning
0b0	When read, means the cycle counter is disabled. When written, has no effect.
0b1	When read, means the cycle counter is enabled. When written, enables the cycle counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter enable bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 is disabled. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 event counter is enabled. When written, enables PMU.PMEVCNTR<n>_EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMCNTENSET_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xC00

PMCNTENSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC00

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xC00

PMCNTENSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC00	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xC04

PMCNTENSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC04	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCR_EL0, Performance Monitors Control Register

The PMCR_EL0 characteristics are:

Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configuration

External register PMCR_EL0 bits [7:0] are architecturally mapped to AArch32 System register [PMCR\[7:0\]](#).

External register PMCR_EL0 bits [7:0] are architecturally mapped to AArch64 System register [PMCR_EL0\[7:0\]](#).

External register PMCR_EL0 bits [7:0] are architecturally mapped to AArch32 System register [PMCR\[7:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMCR_EL0 are RES0.

PMCR_EL0 is in the Core power domain.

This register is only partially mapped to the internal [PMCR](#) System register. An external agent must use other means to discover the information held in [PMCR\[31:11\]](#), such as accessing [PMU.PMCFGR](#) and the ID registers. [PMCFGR](#) and the ID registers.

Attributes

PMCR_EL0 is a [32-bit register](#).

- 64-bit register when FEAT_PMUv3_EXT64 is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented:

63	31	62	30	61	29	60	28	59	27	58	26	57	25	56	24	55	23	54	22	53	21	52	20	51	19	50	18	49	17	48	16	47	15	46	14	45	13	44	12	43	11	42	10	41
RAZ/WI																	RES0															RES0FZ					FZ					9		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9																						

Bits [63:33]

Reserved, RES0.

FZS, bit [32]

When FEAT_SPEv1p2 is implemented:

Freeze-on-SPE event. Stop counters when [PMBLIMITR_EL1](#).{PMFZ,E} == {1,1} and [PMBSR_EL1](#).S == 1.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.

- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is [PMCR_EL0.N](#).

FZS	Meaning
0b0	Do not freeze on Statistical Profiling Buffer Management event.
0b1	Event counter PMEVCNTR<n>_EL0 does not count following a Statistical Profiling Buffer Management event if n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - When the implementation only supports execution in AArch64 state, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [31:11]

Reserved, RAZ/WI.

Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

Bit [10]

Reserved, RES0.

FZO, bit [9]

When FEAT_PMUv3p7 is implemented:

Freeze-on-overflow. Stop event counters on overflow.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is [PMCR_EL0.N](#).

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counter PMU.PMEVCNTR<n>_EL0 does not count when PMEVCNTR<n>_EL0 does not count when PMOVSCLR_EL0[(PMN-1):0] is nonzero and n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and [PMU.PMCCNTR_EL0](#). [PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [8]

Reserved, RES0.

LP, bit [7]

When FEAT_PMUv3p5 is implemented:

Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

LP	Meaning
0b0	Event counter overflow on increment that causes unsigned overflow of PMU.PMEVCNTR<n> EL0[31:0] . PMEVCNTR<n> EL0[31:0] .
0b1	Event counter overflow on increment that causes unsigned overflow of PMU.PMEVCNTR<n> EL0[63:0] . PMEVCNTR<n> EL0[63:0] .

If PMN is not 0, this bit affects the operation of event counters in the range [0 .. (PMN-1)].

The field does not affect the operation of other event counters and [PMU.PMCCNTR_EL0](#).[PMCCNTR_EL0](#).

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LC, bit [6]

When AArch32 is supported:

Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit.

LC	Meaning
0b0	Cycle counter overflow on increment that causes unsigned overflow of PMU.PMCCNTR_EL0[31:0] . PMCCNTR_EL0[31:0] .
0b1	Cycle counter overflow on increment that causes unsigned overflow of PMU.PMCCNTR_EL0[63:0] . PMCCNTR_EL0[63:0] .

Arm deprecates use of [PMU.PMCR_EL0.LC = 0](#).[PMCR_EL0.LC = 0](#).

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

DP, bit [5]**When EL3 is implemented or (FEAT_PMUv3p1 is implemented and EL2 is implemented):**

Disable cycle counter when event counting is prohibited. The possible values of this bit are:

DP	Meaning
0b0	Cycle counting by PMU.PMCCNTR_EL0 is not affected by this mechanism. PMCCNTR_EL0 is not affected by this mechanism.
0b1	Cycle counting by PMU.PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen. PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD is 1, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL2. PMCCNTR_EL0 is disabled at EL2. If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL3. PMCCNTR_EL0 is disabled at EL3. If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMU.PMCCNTR_EL0 is disabled. PMCCNTR_EL0 is disabled. If EL3 is implemented, MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or MDCR_EL3.MPMX is 0, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL3 and in Secure state. PMCCNTR_EL0 is disabled at EL3 and in Secure state. <p>If MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(MDCR_EL2.HPMN-1)] is prohibited or frozen.</p>

For more information, see 'Prohibiting event and cycle counting'.

The reset behavior of this field is:

- On a Warm reset:
 - When the implementation only supports execution in AArch32 state, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

X, bit [4]**When the implementation includes a PMU event export bus:**

Enable export of events in an IMPLEMENTATION DEFINED PMU event export bus.

X	Meaning
0b0	Do not export events.
0b1	Export events where not prohibited.

This field enables the exporting of events over an IMPLEMENTATION DEFINED PMU event export bus to another device, for example to an OPTIONAL trace unit.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

Otherwise:

Reserved, RAZ/WI.

D, bit [3]

When AArch32 is supported:

Clock divider.

D	Meaning
0b0	When enabled, PMU.PMCCNTR_EL0 counts every clock cycle. PMCCNTR_EL0 counts every clock cycle.
0b1	When enabled, PMU.PMCCNTR_EL0 counts once every 64 clock cycles. PMCCNTR_EL0 counts once every 64 clock cycles.

If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

Arm deprecates use of PMCR_EL0.D = 1.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

Otherwise:

Reserved, RES0.

C, bit [2]

Cycle counter reset. The effects of writing to this bit are:

C	Meaning
0b0	No action.
0b1	Reset PMU.PMCCNTR_EL0 to zero. PMCCNTR_EL0 to zero.

Note

Resetting PMU.PMCCNTR_EL0 does not change the cycle counter overflow bit. If PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset.

Access to this field is **WO/RAZ**.

P, bit [1]

Event counter reset. The effects of writing to this bit are:

P	Meaning
0b0	No action.
0b1	Reset all event counters, not including PMU.PMCCNTR_EL0 , to zero. PMCCNTR_EL0 , to zero.

Note

Resetting the event counters does not change the event counter overflow bits. If FEAT_PMuV3p5 is implemented, the value of [MDCR_EL2.HLP](#), or PMCR_EL0.LP is ignored and bits [63:0] of all affected event counters are reset.

Access to this field is **WO/RAZ**.

E, bit [0]

Enable.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is PMCR_EL0.N.

E	Meaning
0b0	PMCCNTR_EL0 is disabled and event counters PMEVCNTR<n>_EL0 PMU.PMCCNTR_EL0 is disabled and event counters PMU.PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are disabled.
0b1	PMCCNTR_EL0 and event counters PMEVCNTR<n>_EL0 , where n is in the range of affected event counters, are enabled by PMCNTESET_EL0 PMU.PMCCNTR_EL0 and event counters PMU.PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by PMU.PMCNTESET_EL0.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMCR_EL0**Note**

[SoftwareLockStatus\(\)](#) depends on the type of access attempted and [AllowExternalPMUAccess\(\)](#) has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCR_EL0 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
RAZ/WI																						RES0FZORES0LP LCDP X D C P E															
Component											Offset											Instance															
PMU											0xE04											PMCR_EL0															

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Bits [31:11]

Reserved, RAZ/WI.

Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

Bit [10]

Reserved, RES0.

FZO, bit [9]

When FEAT_PMUv3p7 is implemented:

Freeze-on-overflow. Stop event counters on overflow.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is PMCR_EL0.N.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counter PMU.PMEVCNTR<n>_EL0 does not count when PMOVSCLR_EL0 [(PMN-1):0] is nonzero and n is in the range of affected event counters.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters and PMU.PMCCNTR_EL0.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [8]

Reserved, RES0.

LP, bit [7]

When FEAT_PMUv3p5 is implemented:

Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is PMCR_EL0.N.

LP	Meaning
0b0	Event counter overflow on increment that causes unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0].
0b1	Event counter overflow on increment that causes unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0].

If PMN is not 0, this bit affects the operation of event counters in the range [0 .. (PMN-1)].

The field does not affect the operation of other event counters and PMU.PMCCNTR_EL0.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

LC, bit [6]

When AArch32 is supported:

Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit.

LC	Meaning
0b0	Cycle counter overflow on increment that causes unsigned overflow of PMU.PMCCNTR_EL0[31:0].
0b1	Cycle counter overflow on increment that causes unsigned overflow of PMU.PMCCNTR_EL0[63:0].

Arm deprecates use of PMU.PMCR_EL0.LC = 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES1.

DP, bit [5]

When EL3 is implemented or (FEAT_PMUv3p1 is implemented and EL2 is implemented):

Disable cycle counter when event counting is prohibited. The possible values of this bit are:

DP	Meaning
0b0	Cycle counting by PMU.PMCCNTR_EL0 is not affected by this mechanism.
0b1	<p>Cycle counting by PMU.PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen:</p> <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD is 1, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL2. If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL3. If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMU.PMCCNTR_EL0 is disabled. If EL3 is implemented, MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or MDCR_EL3.MPMX is 0, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL3 and in Secure state. <p>If MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(MDCR_EL2.HPMN-1)] is prohibited or frozen.</p>

For more information, see 'Prohibiting event and cycle counting'.

The reset behavior of this field is:

- On a Warm reset:
 - When the implementation only supports execution in AArch32 state, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

X, bit [4]

When the implementation includes a PMU event export bus:

Enable export of events in an IMPLEMENTATION DEFINED PMU event export bus.

X	Meaning
0b0	Do not export events.
0b1	Export events where not prohibited.

This field enables the exporting of events over an IMPLEMENTATION DEFINED PMU event export bus to another device, for example to an OPTIONAL trace unit.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

Otherwise:

Reserved, RAZ/WI.

D, bit [3]**When AArch32 is supported:**

Clock divider.

D	Meaning
0b0	When enabled, PMU.PMCCNTR_EL0 counts every clock cycle.
0b1	When enabled, PMU.PMCCNTR_EL0 counts once every 64 clock cycles.

If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

Arm deprecates use of PMCR_EL0.D = 1.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

Otherwise:

Reserved, RES0.

C, bit [2]

Cycle counter reset. The effects of writing to this bit are:

C	Meaning
0b0	No action.
0b1	Reset PMU.PMCCNTR_EL0 to zero.

Note

Resetting PMU.PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset.

Access to this field is **WO/RAZ**.

P, bit [1]

Event counter reset. The effects of writing to this bit are:

P	Meaning
0b0	No action.
0b1	Reset all event counters, not including PMU.PMCCNTR_EL0, to zero.

Note

Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the value of MDCR_EL2.HLP, or PMCR_EL0.LP is ignored and bits [63:0] of all affected event counters are reset.

Access to this field is **WO/RAZ**.

E, bit [0]

Enable.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If EL2 is not implemented, PMN is PMCR_EL0.N.

E	Meaning
0b0	PMU.PMCCNTR_EL0 is disabled and event counters PMU.PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are disabled.
0b1	PMU.PMCCNTR_EL0 and event counters PMU.PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by PMU.PMCNTENSET_EL0.

If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].

This field does not affect the operation of other event counters.

The operation of this field applies even when EL2 is disabled in the current Security state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing PMCR_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xE04

PMCR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE04

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xE10

PMCR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE10

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMDEVAFF, Performance Monitors Device Affinity register

The PMDEVAFF characteristics are:

Purpose

Copy of the PE [MPIDR_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configuration

This register is present only when FEAT_PMUv3_EXT64 is implemented. Otherwise, direct accesses to PMDEVAFF are RES0.

Attributes

PMDEVAFF is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
MPIDR_EL1																															
MPIDR_EL1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MPIDR_EL1, bits [63:0]

[MPIDR_EL1](#). Read-only copy of [MPIDR_EL1](#), as seen from the highest implemented Exception level.

Accessing PMDEVAFF

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFA8

PMDEVAFF can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFA8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMDEVAFF0, Performance Monitors Device Affinity register 0

The PMDEVAFF0 characteristics are:

Purpose

Copy of the low half of the PE [MPIDR_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMDEVAFF0 are RES0.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required if the external interface to the PMU is implemented.

Attributes

PMDEVAFF0 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPIDR_EL1lo																															

MPIDR_EL1lo, bits [31:0]

[MPIDR_EL1](#) low half. Read-only copy of the low half of [MPIDR_EL1](#), as seen from the highest implemented Exception level.

Accessing PMDEVAFF0

PMDEVAFF0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFA8	PMDEVAFF0

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMDEVAFF0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFA8

PMDEVAFF0 can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xFA8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMDEVAFF1, Performance Monitors Device Affinity register 1

The PMDEVAFF1 characteristics are:

Purpose

Copy of the high half of the PE [MPIDR_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configuration

This register is present only when FEAT_PMuV3_EXT32 is implemented. Otherwise, direct accesses to PMDEVAFF1 are RES0.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required if the external interface to the PMU is implemented.

Attributes

PMDEVAFF1 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPIDR_EL1hi																															

MPIDR_EL1hi, bits [31:0]

[MPIDR_EL1](#) high half. Read-only copy of the high half of [MPIDR_EL1](#), as seen from the highest implemented Exception level.

Accessing PMDEVAFF1

PMDEVAFF1 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFAC	PMDEVAFF1

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMDEVAFF1

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFAC

PMDEVAFF1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFAC

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMDEVARCH, Performance Monitors Device Architecture register

The PMDEVARCH characteristics are:

Purpose

Identifies the programmers' model architecture of the Performance Monitor component.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMDEVARCH are RES0.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

PMDEVARCH is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHITECT											PRESENT	REVISION				ARCHVER				ARCHPART											

ARCHITECT, bits [31:21]

Defines the architecture of the component. For Performance Monitors, this is Arm Limited.

Bits [31:28] are the JEP106 continuation code, 0x4.

Bits [27:21] are the JEP106 ID code, 0x3B.

Reads as 0b01000111011.

Access to this field is **RO**.

PRESENT, bit [20]

Indicates that the DEVARCH is present.

Reads as 0b1.

Access to this field is **RO**.

REVISION, bits [19:16]

Defines the architecture revision. For architectures defined by Arm this is the minor revision.

For Performance Monitors, the revision defined by Armv8 is 0x0.

All other values are reserved.

Reads as 0b0000.

Access to this field is **RO**.

ARCHVER, bits [15:12]

Architecture Version. Defines the architecture version of the component.

ARCHVER	Meaning
0b0010	Performance Monitors Extension version 3, PMUv3.

All other values are reserved.

PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHVER is PMDEVARCH.ARCHID[15:12].

Access to this field is **RO**.

ARCHPART, bits [11:0]

Architecture Part. Defines the architecture of the component.

ARCHPART	Meaning	Applies when
0xA16	Armv8-A PE performance monitors.	
0xA26	Armv8-A PE performance monitors, including the 64-bit programmers' model extension.	From Armv8.8

PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHPART is PMDEVARCH.ARCHID[11:0].

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMDEVARCH

PMDEVARCH can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFBC	PMDEVARCH

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMDEVARCH

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFBC

PMDEVARCH can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFBC

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMDEVID, Performance Monitors Device ID register

The PMDEVID characteristics are:

Purpose

Provides information about features of the Performance Monitors implementation.

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMDEVID are RES0.

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required from Armv8.2 and in any implementation that includes FEAT_PCSRv8p2. Otherwise, its location is RES0.

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID.PCSample](#).

Attributes

PMDEVID is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
												RES0														PMSS		PCSample		PCSample			

Bits [31:84]

Reserved, RES0.

PMSS, bits [7:4]

PMU Snapshot extension. Defined values are:

PMSS	Meaning
0b0000	PMU snapshot extension not implemented.
0b0001	PMU snapshot extension implemented.

All other values are reserved.

FEAT_PMUv3_SS implements the functionality identified by the value 0b0001.

PCSample	Meaning
0b0000	PC Sample-based Profiling Extension is not implemented in the Performance Monitors register space.
0b0001	PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.
0b0010	As 0b0001, and adds support for PMPCSTL.

FEAT PCSRv8p2 implements the functionality identified by the value 0b0001.

FEAT PCSRv8p9 implements the functionality identified by the value 0b0010.

If FEAT_PCSRv8p2 is not implemented, then the only permitted value is 0b0000.

From Armv8.2, when FEAT PCSRv8p2 is implemented, the value 0b0000 is not permitted.

From Armv8.9, when FEAT PCSRv8p9 is implemented, the value 0b0001 is not permitted.

Accessing PMDEVID

PMDEVID can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFC8	PMDEVID

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMDEVID

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFC8

PMDEVID can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFC8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

(old)

htmldiff from-

(new)

PMDEVTYPE, Performance Monitors Device Type register

The PMDEVTYPE characteristics are:

Purpose

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configuration

Implementation of this register is OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

PMDEVTYPE is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RES0												SUB				MAJOR			

Bits [31:8]

Reserved, RES0.

SUB, bits [7:4]

Subtype. Indicates this is a component within a PE.

Reads as 0b0001.

Access to this field is **RO**.

MAJOR, bits [3:0]

Major type. Indicates this is a performance monitor component.

Reads as 0b0110.

Access to this field is **RO**.

Accessing PMDEVTYPE

PMDEVTYPE can be accessed through the external debug interface:

Component	Offset	Instance
-----------	--------	----------

	PMU		0xFCC		PMDEVTYPE
--	-----	--	-------	--	-----------

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMDEVTYPE

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFCC

PMDEVTYPE can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFCC

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The PMEVCNTR<n>_EL0 characteristics are:

Purpose

Holds event counter **<n>**, which counts events, where **<n>** is 0 to 30.

Configuration

External register `PMEVCNTR<n>_EL0` bits [31:0] are architecturally mapped to AArch64 System register [PMEVCNTR<n>_EL0\[31:0\]](#).

External register `PMEVCNTR<n>_EL0` bits [63:32] are architecturally mapped to AArch64 System register `PMEVCNTR<n>_EL0[63:32]` when FEAT_PMUv3p5 is implemented.

External register PMEVCNTR<n>_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMEVCNTR<n>\[31:0\]](#).

This register is present only when FEAT_PMuV3_EXT is implemented. Otherwise, direct accesses to PMEVCNTR<n> EL0 are RES0.

PMEVCNTR<n>>_EL0 is in the Core power domain.

Attributes

PMEVCNTR<n>>_EL0 is a:

- 64-bit register when FEAT_PMUv3p5 is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3p5 is implemented:

63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32

Event counter n

Event counter n

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Bits [63:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

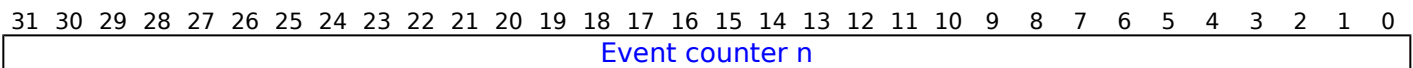
If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the [PMCR.LP](#) and [HDCR.HLP](#) bits are RAZ/WI, then locations in the external interface to the performance monitors that map to `PMEVCNTR<n> EL0[63:32]` return UNKNOWN values on reads.

If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:



Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVCNTR<n>_EL0

External accesses to the performance monitors ignore the following controls:

- [PMUSERENR_EL0](#).
- If implemented, [MDCR_EL2](#).{TPM, TPMCR, HPMN}.
- [MDCR_EL3](#).TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMuV3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMEVCNTR<n>_EL0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x000 + (8 * n)	PMEVCNTR<n>_EL0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Accessing PMEVCNTR<n>_EL0

External accesses to the performance monitors ignore the following controls:

- [PMUSERENR_EL0](#).
- If implemented, [MDCR_EL2](#).{TPM, TPMCR, HPMN}.
- [MDCR_EL3](#).TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMuV3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

BlockAccess at address $0x000 + (8 * n)$

Frame	Offset
PMU	0x000 + (8 * n)

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

no old file

htmldiff from-

(new)

PMEVCNTSVR<n>_EL1, Performance Monitors Event Count Saved Value Register <n>, n = 0 - 30

The PMEVCNTSVR<n>_EL1 characteristics are:

Purpose

Captures the PMU Event counter <n>, PMU.PMEVCNTR<n>_EL0.

Configuration

External register PMEVCNTSVR<n>_EL1 bits [63:0] are architecturally mapped to AArch64 System register [PMEVCNTSVR<n>_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMEVCNTSVR<n>_EL1 are RES0.

PMEVCNTSVR<n>_EL1 is in the Core power domain.

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

PMEVCNTSVR<n>_EL1 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																EVCNT															
																EVCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EVCNT, bits [63:0]

Sampled Event Count. The value of PMU.PMEVCNTR<n>_EL0 at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVCNTSVR<n>_EL1

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address $0x600 + (8 * n)$

PMEVCNTSVR<n>_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	$0x600 + (8 * n)$

- When !AllowExternalPMSSAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMEVTYPEPER<n>_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>_EL0 characteristics are:

Purpose

Configures event counter <n>, where <n> is 0 to 30.

Configuration

External register PMEVTYPER<n>_EL0 bits [63:0] are architecturally mapped to AArch64 System register [PMEVTYPER<n>_EL0\[63:0\]](#).

External register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMEVTYPER<n>\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMEVTYPER<n>_EL0 are RES0.

PMEVTYPER<n>_EL0 is in the Core power domain.

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

PMEVTYPER<n>_EL0 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60		59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32		
TC			TERES0		RES0		TH		SYNC		RES0													TH										
P	U	NSK	NSU		NSH		M		MT	SH	T	RLK	RLU	RLH	RES0			evtCount[15:10]					evtCount[9:0]											
31	30	29	28		27		26		25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TC, bits [63:61]

When FEAT_PMUv3_EDGEFEAT_PMUv3_TH is implemented and PMU.PMEVTYPER<n>_EL0.TE == 1:

Threshold Control. Defines the threshold function. In the description of this field, the value V is the value the event specified by PMEVTYPER<n>_EL0.evtCount would increment the counter by on a processor cycle if the threshold function is disabled. Comparisons treat V and this PMEVTYPER<n>_EL0.TH as unsigned integer values.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMU.PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>.TH.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b110	Less-than. The counter increments by V on each processor cycle when V is less than $PMEVTYPER<n>_{EL0}.TH$.
0b111	Less-than, count. The counter increments by 1 on each processor cycle when V is less than $PMEVTYPER<n>_{EL0}.TH$.
0b0010b000	EqualNot-equal, to not-equal. The counter increments by 1V on each processor cycle when V is not equal to $PMEVTYPER<n>_{EL0}.TH$. If $PMEVTYPER<n>_{EL0}.TH$ is zero, the threshold function is disabled. B is not equal to TH and V_B was equal to TH on the previous processor cycle.
0b0100b001	EqualNot-equal, to/from count, not-equal. The counter increments by 1 on each processor cycle when either: V is not equal to $PMEVTYPER<n>_{EL0}.TH$. <ul style="list-style-type: none"> V_B is not equal to TH and V_B was equal to TH on the previous processor cycle. V_B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b0110b010	Not-equalEquals, to equal. The counter increments by 1V on each processor cycle when V is equal to $PMEVTYPER<n>_{EL0}.TH$. B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b1010b011	Less-thanEquals, to count, greater-than-or-equal. The counter increments by 1 on each processor cycle when V is equal to $PMEVTYPER<n>_{EL0}.TH$. B is greater than or equal to TH and V_B was less than TH on the previous processor cycle.
0b1100b100	Less-thanGreater-than-or-equal, to/from greater-than-or-equal. The counter increments by 1V on each processor cycle when either: V is $PMEVTYPER<n>_{EL0}.TH$ or more. <ul style="list-style-type: none"> V_B is greater than or equal to TH and V_B was less than TH on the previous processor cycle. V_B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.
0b1110b101	Greater-than-or-equal, to count, less-than. The counter increments by 1 on each processor cycle when V is $PMEVTYPER<n>_{EL0}.TH$ or more. B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

When ((FEAT_PMuV3_EDGE is implemented and $PMU.PMEVTYPER<n>_{EL0}.TE == 0$) or FEAT_PMuV3_EDGE is not implemented) and FEAT_PMuV3_TH is implemented:

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by $PMU.PMEVTYPER<n>_{EL0}$ would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of $PMEVTYPER<n>_{EL0}.TH$.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b000	Not-equal. The counter increments by V_B on each processor cycle when V_B is not equal to TH. If TH is zero, the threshold function is disabled.
0b001	Not-equal, count. The counter increments by 1 on each processor cycle when V_B is not equal to TH.
0b010	Equals. The counter increments by V_B on each processor cycle when V_B is equal to TH.
0b011	Equals, count. The counter increments by 1 on each processor cycle when V_B is equal to TH.
0b100	Greater-than-or-equal. The counter increments by V_B on each processor cycle when V_B is greater than or equal to TH.
0b101	Greater-than-or-equal, count. The counter increments by 1 on each processor cycle when V_B is greater than or equal to TH.
0b110	Less-than. The counter increments by V_B on each processor cycle when V_B is less than TH.
0b111	Less-than, count. The counter increments by 1 on each processor cycle when V_B is less than TH.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMU.PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>.TH.

Comparisons treat V_B and TH as unsigned integer values.

TE, bit [60]

When FEAT_PMUv3_EDGE is implemented:

Threshold Edge. Enables the edge condition. When PMEVTYPER<n>.TE is 1, the event counter increments on cycles when the result of the threshold condition changes. See PMEVTYPER<n>.TC for more information.

TE	Meaning
0b0	Threshold edge condition disabled.
0b1	Threshold edge condition enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

BitBits [5960:44]

Reserved, RES0.

SYNC, bit [58]**When FEAT_SEBEP is implemented:**

Synchronous Mode. Controls whether a PMU exception generated by the counter is synchronous or asynchronous.

SYNC	Meaning
0b0	Asynchronous PMU exception is enabled.
0b1	Synchronous PMU exception is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [57:44]

Reserved, RES0.

TH, bits [43:32]**When FEAT_PMUv3_TH is implemented:**

Threshold value. Provides the unsigned value for the threshold function defined by PMEVTYPER<n>_EL0.TC.

If PMEVTYPER<n>_EL0.TC is 0b000 and PMEVTYPER<n>_EL0.TH is zero, then the threshold function is disabled.

If `PMMIR_EL1.PMMIR.THWIDTH` is less than 12, then bits `PMEVTYPER<n>_EL0.TH[11:PMMIR_EL1.PMMIR.THWIDTH]` are RES0. This accounts for the behavior when writing a value greater-than-or-equal-to $2^{(PMMIR_EL1.THWIDTH - PMMIR.THWIDTH)}$.

The reset behavior of this field is:

- On a Warm reset:
 - When AArch32 is supported, this field resets to 0.
 - Otherwise, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

P, bit [31]

Privileged filtering bit. Controls counting in EL1.

If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit.

If FEAT_RME is implemented, then counting in Realm EL1 is further controlled by the PMEVTYPER<n>_EL0.RLK bit.

P	Meaning
0b0	Count events in EL1.
0b1	Do not count events in EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

User filtering bit. Controls counting in EL0.

If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit.

If FEAT_RME is implemented, then counting in Realm EL0 is further controlled by the PMEVTYPER<n>_EL0.RLU bit.

U	Meaning
0b0	Count events in EL0.
0b1	Do not count events in EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 (Hypervisor) filtering bit. Controls counting in EL2.

If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit.

If FEAT_RME is implemented, then counting in Realm EL2 is further controlled by the PMEVTYPER<n>_EL0.RLH bit.

NSH	Meaning
0b0	Do not count events in EL2.
0b1	Count events in EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

M, bit [26]

When EL3 is implemented:

EL3 filtering bit.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted.

Otherwise, events in EL3 are not counted.

Most applications can ignore this field and set its value to 0b0.

Note

This field is not visible in the AArch32 [PMEVTYPER<n>](#) System register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

MT, bit [25]

When (FEAT_MTPMU is implemented and enabled) or an IMPLEMENTATION DEFINED multi-threaded PMU Extension is implemented:

Multithreading.

MT	Meaning
0b0	Count events only on controlling PE.
0b1	Count events from any PE with the same affinity at level 1 and above as this PE.

Note

- When the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach, an implementation is described as multi-threaded. That is, the performance of PEs at the lowest affinity level is highly interdependent.
- Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SH, bit [24]**When FEAT_SEL2 is implemented and EL3 is implemented:**

Secure EL2 filtering.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted.

Otherwise, events in Secure EL2 are not counted.

Note

This field is not visible in the AArch32 [PMEVTYPER<n>](#) System register.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

T, bit [23]**When FEAT_TME is implemented:**

Transactional state filtering bit. Controls counting of Attributable events in Non-transactionalTransactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count Attributable events in Non-transactionalTransactional state.

For each Unattributable event, it is IMPLEMENTATION DEFINED whether the filtering applies.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]**When FEAT_RME is implemented:**

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted.

Otherwise, events in Realm EL1 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]**When FEAT_RME is implemented:**

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]**When FEAT_RME is implemented:**

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted.

Otherwise, events in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:16]

Reserved, RES0.

evtCount[15:10], bits [15:10]**When FEAT_PMUv3p1 is implemented:**

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

evtCount[9:0], bits [9:0]

Event to count. The event number of the event that is counted by event counter

PMU.PMEVCNTR<n>_EL0.PMEVCNTR<n>_EL0.

Software must program this field with an event that is supported by the PE being programmed.

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

Note

Arm recommends this behavior for all implementations of FEAT_PMUv3.

Otherwise, if PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is UNKNOWN.

Note

UNPREDICTABLE means the event must not expose privileged information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMEVTYPER<n>_EL0

If FEAT_PMUv3_TH or FEAT_PMUv3p8 is implemented, bits [63:32] of this interface are accessible at offset 0xA00 + (4*n). Otherwise accesses at this offset are IMPLEMENTATION DEFINED.

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMEVTYPER<n>_EL0 can be accessed through the external debug interface:

Component	Offset	Instance	Range
PMU	0x400 + (4 * n)	PMEVTYPER<n>_EL0	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
-----------	--------	----------	-------

PMU	0xA00 + (4 * n)	PMEVTYPEPER<n>_EL0	63:32
-----	-----------------	--------------------	-------

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

Accessing PMEVTYPER<n>_EL0

If FEAT_PMUv3_TH is implemented, and at least one of FEAT_PMUv3_TH or FEAT_PMUv3p8 is implemented, bits [63:32] of this interface are accessible at offset 0xA00 + (4*n). Otherwise accesses at this offset are IMPLEMENTATION DEFINED.

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0x400 + (4 * n)

PMEVTYPEPER<n>_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0x400 + (4 * n)	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0x400 + (8 * n)

PMEVTYPEPER<n>_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0x400 + (8 * n)	63:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xA00 + (4 * n)

PMEVTYPEPER<n>_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xA00 + (4 * n)	63:32

- When DoubleLockStatus(), or !IsCorePowered(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.

- When SoftwareLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register are **RW**.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165fb91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

no old file

htmldiff from-

(new)

PMICFILTR_EL0, Performance Monitors Instruction Counter Filter Register

The PMICFILTR_EL0 characteristics are:

Purpose

Configures the Instruction Counter.

Configuration

External register PMICFILTR_EL0 bits [63:0] are architecturally mapped to AArch64 System register [PMICFILTR_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3_ICNTR is implemented. Otherwise, direct accesses to PMICFILTR_EL0 are RES0.

PMICFILTR_EL0 is in the Core power domain.

Note

If FEAT_Debugv8p4 is implemented, the OPTIONAL Software Lock is not implemented.

Attributes

PMICFILTR_EL0 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0					SYNC	RES0																									
P	U	NSK	NSU	NSH	M	RES0	SH	T	RLK	RLU	RLH	RES0				evtCount															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:59]

Reserved, RES0.

SYNC, bit [58]

When FEAT_SEBEP is implemented:

Synchronous mode. Controls whether a PMU exception generated by the counter is synchronous or asynchronous.

SYNC	Meaning
0b0	Asynchronous PMU exception is enabled.
0b1	Synchronous PMU exception is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [57:32]

Reserved, RES0.

P, bit [31]

EL1 filtering. Controls counting instructions in EL1.

P	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in EL1 are not counted.

If EL3 is implemented, then counting instructions in Non-secure EL1 is further controlled by PMICFILTR_EL0.NSK, and counting instructions in EL3 is further controlled by PMICFILTR_EL0.M.

If FEAT_RME is implemented, then counting instructions in Realm EL1 is further controlled by PMICFILTR_EL0.RLK.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

U, bit [30]

EL0 filtering. Controls counting instructions in EL0.

U	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in EL0 are not counted.

If EL3 is implemented, then PMICFILTR_EL0.NSU further controls filtering instructions in Non-secure EL0.

If FEAT_RME is implemented, then counting instructions in Realm EL1 is further controlled by PMICFILTR_EL0.RLU.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

NSK, bit [29]**When EL3 is implemented:**

Non-secure EL1 filtering. Controls counting instructions in Non-secure EL1. If PMICFILTR_EL0.NSK is equal to PMICFILTR_EL0.P, then instructions in Non-secure EL1 are counted. Otherwise, instructions in Non-secure EL1 are not counted.

NSK	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in Non-secure EL1 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in Non-secure EL1 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSU, bit [28]**When EL3 is implemented:**

Non-secure EL0 filtering. Controls counting instructions in Non-secure EL0. If PMICFILTR_EL0.NSU is equal to PMICFILTR_EL0.U, then instructions in Non-secure EL0 are counted. Otherwise, instructions in Non-secure EL0 are not counted.

NSU	Meaning
0b0	If PMICFILTR_EL0.U == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.U == 1, then instructions in Non-secure EL0 are not counted.
0b1	If PMICFILTR_EL0.U == 0, then instructions in Non-secure EL0 are not counted. If PMICFILTR_EL0.U == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSH, bit [27]**When EL2 is implemented:**

EL2 filtering. Controls counting instructions in EL2.

NSH	Meaning
0b0	Instructions in EL2 are not counted.
0b1	This bit has no effect on filtering of instructions.

If EL3 is implemented and FEAT_SEL2 is implemented, then counting instructions in Secure EL2 is further controlled by PMICFILTR_EL0.SH.

If FEAT_RME is implemented, then counting instructions in Realm EL2 is further controlled by PMICFILTR_EL0.RLH.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

M, bit [26]**When EL3 is implemented:**

EL3 filtering. Controls counting instructions in EL3. If PMICFILTR_EL0.M is equal to PMICFILTR_EL0.P, then instructions in EL3 are counted. Otherwise, instructions in EL3 are not counted.

M	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in EL3 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in EL3 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bit [25]

Reserved, RES0.

SH, bit [24]

When EL3 is implemented and FEAT_SEL2 is implemented:

Secure EL2 filtering. Controls counting instructions in Secure EL2. If PMICFILTR_EL0.SH is not equal to PMICFILTR_EL0.NSH, then instructions in Secure EL2 are counted. Otherwise, instructions in Secure EL2 are not counted.

SH	Meaning
0b0	If PMICFILTR_EL0.NSH == 0, then instructions in Secure EL2 are not counted. If PMICFILTR_EL0.NSH == 1, then this bit has no effect on filtering of instructions.
0b1	If PMICFILTR_EL0.NSH == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.NSH == 1, then instructions in Secure EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When !IsSecureEL2Enabled(), access to this field is **RES0**.
- Otherwise, access to this field is **RW**.

Otherwise:

Reserved, RES0.

T, bit [23]

When FEAT_TME is implemented:

Non-transactional state filtering. Controls counting instructions in Non-transactional state.

T	Meaning
0b0	This bit has no effect on filtering of instructions.
0b1	Instructions in Non-transactional state are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLK, bit [22]

When FEAT_RME is implemented:

Realm EL1 filtering. Controls counting instructions in Realm EL1. If PMICFILTR_EL0.RLK is equal to PMICFILTR_EL0.P, then instructions in Realm EL1 are counted. Otherwise, instructions in Realm EL1 are not counted.

RLK	Meaning
0b0	If PMICFILTR_EL0.P == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.P == 1, then instructions in Realm EL1 are not counted.
0b1	If PMICFILTR_EL0.P == 0, then instructions in Realm EL1 are not counted. If PMICFILTR_EL0.P == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLU, bit [21]

When FEAT_RME is implemented:

Realm EL0 filtering. Controls counting instructions in Realm EL0. If PMICFILTR_EL0.RLU is equal to PMICFILTR_EL0.U, then instructions in Realm EL0 are counted. Otherwise, instructions in Realm EL0 are not counted.

RLU	Meaning
0b0	If PMICFILTR_EL0.U == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.U == 1, then instructions in Realm EL0 are not counted.
0b1	If PMICFILTR_EL0.U == 0, then instructions in Realm EL0 are not counted. If PMICFILTR_EL0.U == 1, then this bit has no effect on filtering of instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

RLH, bit [20]**When FEAT_RME is implemented:**

Realm EL2 filtering. Controls counting instructions in Realm EL2. If PMICFILTR_EL0.RLH is not equal to PMICFILTR_EL0.NSH, then instructions in Realm EL2 are counted. Otherwise, instructions in Realm EL2 are not counted.

RLH	Meaning
0b0	If PMICFILTR_EL0.NSH == 0, then instructions in Realm EL2 are not counted. If PMICFILTR_EL0.NSH == 1, then this bit has no effect on filtering of instructions.
0b1	If PMICFILTR_EL0.NSH == 0, then this bit has no effect on filtering of instructions. If PMICFILTR_EL0.NSH == 1, then instructions in Realm EL2 are not counted.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [19:16]

Reserved, RES0.

evtCount, bits [15:0]

Event to count.

Reads as 0x0008.

Access to this field is **RO**.

Accessing PMICFILTR_EL0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x480

PMICFILTR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x480

- When DoubleLockStatus(), or !IsCorePowered(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.
- When SoftwareLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register are **RW**.

no old file

htmldiff from-

(new)

PMICNTR_EL0, Performance Monitors Instruction Counter Register

The PMICNTR_EL0 characteristics are:

Purpose

If event counting is not prohibited and the instruction counter is enabled, the counter increments for each architecturally-executed instruction, according to the configuration specified by PMU.PMICFILTR_EL0.

Configuration

External register PMICNTR_EL0 bits [63:0] are architecturally mapped to AArch64 System register [PMICNTR_EL0\[63:0\]](#).

This register is present only when FEAT_PMUv3_ICNTR is implemented. Otherwise, direct accesses to PMICNTR_EL0 are RES0.

PMICNTR_EL0 is in the Core power domain.

Attributes

PMICNTR_EL0 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ICNT															
																ICNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ICNT, bits [63:0]

Instruction Counter.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMICNTR_EL0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x100

PMICNTR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x100

- When DoubleLockStatus(), or !IsCorePowered(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.

- When SoftwareLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

PMICNTSVR_EL1, Performance Monitors Instruction Count Saved Value Register

The PMICNTSVR_EL1 characteristics are:

Purpose

Captures the PMU Instruction counter, PMU.PMICNTR_EL0.

Configuration

External register PMICNTSVR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [PMICNTSVR_EL1\[63:0\]](#).

This register is present only when FEAT_PMUv3_ICNTR is implemented and FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMICNTSVR_EL1 are RES0.

Attributes

PMICNTSVR_EL1 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																ICNT															
																ICNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ICNT, bits [63:0]

Sampled Instruction Count. The value of PMU.PMICNTR_EL0 at the last Capture event.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMICNTSVR_EL1

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x700

PMICNTSVR_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x700

- When !AllowExternalPMSSAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMIIDR, Performance Monitors Implementation Identification Register

The PMIIDR characteristics are:

Purpose

Provides discovery information about the Performance Monitor component.

Configuration

This register is present only when (FEAT_PMUv3_EXT32 is implemented and an implementation implements PMIIDR) or FEAT_PMUv3_EXT64 is implemented. Otherwise, direct accesses to PMIIDR are RES0.

Attributes

PMIIDR is a:

- 64-bit register when FEAT_PMUv3_EXT64 is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32				
RES0																																			
ProductID																Variant				Revision				Implementer[10:7]				RES0		Implementer[6:0]					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

Bits [63:32]

Reserved, RES0.

ProductID, bits [31:20]

Part number, bits [11:0]. The part number is selected by the designer of the component.

Matches the PMU.PMPIDR1.PART_1, PMU.PMPIDR0.PART_0 fields if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Variant, bits [19:16]

Component major revision.

Defines either a variant of the component defined by PMIIDR.ProductID, or the major revision of the component.

When defining a major revision, PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the most significant part.

When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If this field is increased, PMIIDR.Revision should be set to 0b0000.

Matches the PMU.PMPIDR2.REVISION field, if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Component minor revision.

PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the least significant part.

When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If PMIIDR.Variant field is increased, this field should be set to 0b0000, otherwise the value in this field should be increased.

Matches the PMU.PMPIDR3.REVAND field, if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:8, 6:0]

JEDEC-assigned JEP106 identification code of the designer of the component.

Bits [11:8] are the JEP106 bank identifier minus 1 and bits [6:0] are the JEP106 identification code for the designer of the component.

Note

For Arm Limited, the JEP106 bank is 5 and the JEP106 identification code is 0x3B, meaning PMIIDR[11:0] has the value 0x43B.

Bits [11:8] match the PMU.PMPIDR4.DES_2 field, if present.

Bits[6:0] match the {PMPIDR2.DES1, PMPIDR1.DES_0} fields if present.

This field has an IMPLEMENTATION DEFINED value.

The Implementer field is split as follows:

- Implementer[10:7] is PMIIDR[11:8].
- Implementer[6:0] is PMIIDR[6:0].

Access to this field is **RO**.

Bit [7]

Reserved, RES0.

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ProductID												Variant			Revision			Implementer[10:7]			RES0	Implementer[6:0]									

ProductID, bits [31:20]

Part number, bits [11:0]. The part number is selected by the designer of the component.

Matches the PMU.PMPIDR1.PART_1, PMU.PMPIDR0.PART_0 fields if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Variant, bits [19:16]

Component major revision.

Defines either a variant of the component defined by PMIIDR.ProductID, or the major revision of the component.

When defining a major revision, PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the most significant part.

When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If this field is increased, PMIIDR.Revision should be set to 0b0000.

Matches the PMU.PMPIDR2.REVISION field, if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Component minor revision.

PMIIDR.Variant and PMIIDR.Revision together form the revision number of the component, with this field being the least significant part.

When a component is changed, PMIIDR.Variant or PMIIDR.Revision is increased to ensure that software can differentiate between different revisions of the component. If PMIIDR.Variant field is increased, this field should be set to 0b0000, otherwise the value in this field should be increased.

Matches the PMU.PMPIDR3.REVAND field, if present.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:8, 6:0]

JEDEC-assigned JEP106 identification code of the designer of the component.

Bits [11:8] are the JEP106 bank identifier minus 1 and bits [6:0] are the JEP106 identification code for the designer of the component.

Note

For Arm Limited, the JEP106 bank is 5 and the JEP106 identification code is 0x3B, meaning PMIIDR[11:0] has the value 0x43B.

Bits [11:8] match the PMU.PMPIDR4.DES_2 field, if present.

Bits[6:0] match the {PMPIDR2.DES1, PMPIDR1.DES_0} fields if present.

This field has an IMPLEMENTATION DEFINED value.

The Implementer field is split as follows:

- Implementer[10:7] is PMIIDR[11:8].
- Implementer[6:0] is PMIIDR[6:0].

Access to this field is **RO**.

Bit [7]

Reserved, RES0.

Accessing PMIIDR

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xE08

PMIIDR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE08

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMINTEN, Performance Monitors Interrupt Enable register

The PMINTEN characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, PMU.PMCCNTR_EL0, and the event counters PMU.PMEVCNTR<n>_EL0.

Configuration

External register PMINTEN bits [31:0] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[31:0\]](#).

External register PMINTEN bits [63:32] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTEN bits [31:0] are architecturally mapped to AArch32 System register [PMINTENSET\[31:0\]](#).

External register PMINTEN bits [31:0] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[31:0\]](#).

External register PMINTEN bits [63:32] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTEN bits [31:0] are architecturally mapped to AArch32 System register [PMINTENCLR\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT64 is implemented. Otherwise, direct accesses to PMINTEN are RES0.

Attributes

PMINTEN is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

F0, bit [32]

When FEAT_PMUv3_ICNTR is implemented:

Interrupt request on unsigned overflow of PMU.PMICNTR_EL0 enable.

F0	Meaning
0b0	Interrupt request on unsigned overflow of PMU.PMICNTR_ELO disabled.
0b1	Interrupt request on unsigned overflow of PMU.PMICNTR_ELO enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

C, bit [31]

PMU.PMCCNTR_ELO unsigned overflow interrupt request enable bit. Possible values are:

C	Meaning
0b0	The cycle counter overflow interrupt request is disabled.
0b1	The cycle counter overflow interrupt request is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter unsigned overflow interrupt request enable bit for PMU.PMEVCNTR<n>_ELO.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	The PMU.PMEVCNTR<n>_ELO event counter interrupt request is disabled.
0b1	The PMU.PMEVCNTR<n>_ELO event counter interrupt request is enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTEN

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xC50

PMINTEN can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC50

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR_EL1 characteristics are:

Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMU.PMCCNTR_EL0](#), and the event counters [PMU.PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled. [PMCCNTR_EL0](#), and the event counters [PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

Configuration

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[31:0\]](#).

External register PMINTENCLR_EL1 bits [63:32] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENCLR\[31:0\]](#).

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[31:0\]](#).

External register PMINTENCLR_EL1 bits [63:32] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENSET\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMINTENCLR_EL1 are RES0.

PMINTENCLR_EL1 is in the Core power domain.

Attributes

PMINTENCLR_EL1 is a: [32-bit register](#).

- 64-bit register when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented:

63	31	16	23	06	12	96	02	85	92	75	82	65	72	55	62	45	52	35	42	25	32	15	22	05	12	95	02	84	91	74	81	64	71	54	61	44	51	34	41	24	31	14	21	04	11	9
RES0																																														
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9																								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9																								

Bits [63:33]

Reserved, RES0.

F0, bit [32]

When FEAT_PMUv3_ICNTR is implemented:

Interrupt request on unsigned overflow of PMU.PMICNTR_EL0 disable. On writes, allows software to disable the interrupt request on unsigned overflow of PMU.PMICNTR_EL0. On reads, returns the interrupt request on unsigned overflow of PMU.PMICNTR_EL0 enable status.

F0	Meaning
0b0	Interrupt request on unsigned overflow of PMU.PMICNTR_EL0 disabled.
0b1	Interrupt request on unsigned overflow of PMU.PMICNTR_EL0 enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

~~PMCCNTR_EL0~~PMU.PMCCNTR_EL0 overflow interrupt request disable bit. ~~Possible values are:~~

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request disable bit for PMU.PMEVCNTR<n>_EL0. ~~PMEVCNTR<n>_EL0.~~

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI. ~~PMCFGR.N is less than 31, bits [30:PMCFGR.N] are RAZ/WI.~~

P<n>	Meaning
0b0	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect. PMMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, disables the PMU.PMEVCNTR<n>_EL0 interrupt request. PMMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, disables the PMMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENCLR_EL1

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMINTENCLR_EL1 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xC60								PMINTENCLR_EL1															

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

C, bit [31]

PMU.PMCCNTR_EL0 overflow interrupt request disable bit.

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request disable bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, disables the PMU.PMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENCLR_EL1

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xC60

PMINTENCLR_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC60

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xC60

PMINTENCLR_EL1 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC60	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xC64

PMINTENCLR_EL1 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC64	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.

- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMINTENSET_EL1, Performance Monitors Interrupt Enable Set register

The PMINTENSET_EL1 characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMU.PMCCNTR_EL0](#), and the event counters [PMU.PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled. [PMCCNTR_EL0](#), and the event counters [PMEVCNTR<n>_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

Configuration

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[31:0\]](#).

External register PMINTENSET_EL1 bits [63:32] are architecturally mapped to AArch64 System register [PMINTENSET_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENSETI\[31:0\]](#).

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[31:0\]](#).

External register PMINTENSET_EL1 bits [63:32] are architecturally mapped to AArch64 System register [PMINTENCLR_EL1\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PMINTENCLR\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMINTENSET_EL1 are RES0.

PMINTENSET_EL1 is in the Core power domain.

Attributes

PMINTENSET_EL1 is a: [32-bit register](#).

- 64-bit register when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

P<n>	Meaning
0b0	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect. PMMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, enables the PMU.PMEVCNTR<n>_EL0 interrupt request. PMMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, enables the PMMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENSET_EL1

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMINTENSET_EL1 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xC40								PMINTENSET_EL1															

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

C, bit [31]

PMU.PMCCNTR_EL0 overflow interrupt request enable bit.

C	Meaning
0b0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
0b1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow interrupt request enable bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect.
0b1	When read, means that the PMU.PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, enables the PMU.PMEVCNTR<n>_EL0 interrupt request.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMINTENSET_EL1

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xC40

PMINTENSET_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC40

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xC40

PMINTENSET_EL1 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC40	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xC44

PMINTENSET_EL1 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC44	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.

- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMITCTRL, Performance Monitors Integration mode Control register

The PMITCTRL characteristics are:

Purpose

Enables the Performance Monitors to switch from default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

Configuration

Implementation of this register is OPTIONAL.

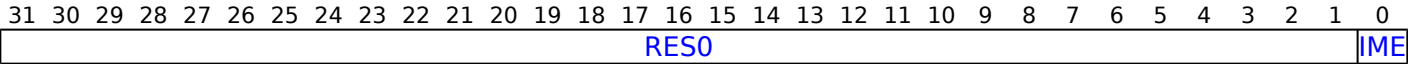
This register is present only when FEAT_PMUv3_EXT32 is implemented and an implementation implements PMITCTRL. Otherwise, direct accesses to PMITCTRL are RES0IMPLEMENTATION DEFINED. whether PMITCTRL is implemented in the Core power domain or in the Debug power domain.

Attributes

PMITCTRL is a 32-bit register.

This register is part of the PMU block.

Field descriptions



Bits [31:1]

Reserved, RES0.

IME, bit [0]

Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED.

IME	Meaning
0b0	Normal operation.
0b1	Integration mode enabled.

The following resets apply:

- If the register is implemented in the Core power domain:
 - On a Cold reset, this field resets to 0.
 - On an External debug reset, the value of this field is unchanged.
 - On a Warm reset, the value of this field is unchanged.
- If the register is implemented in the External debug power domain:
 - On a Cold reset, the value of this field is unchanged.

PMITCTRL can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xF00	PMITCTRL

- When `IsCorePowered()`, `!DoubleLockStatus()`, `!OSLockStatus()` and `SoftwareLockStatus()`, accesses to this register are **RO**.
- When `IsCorePowered()`, `!DoubleLockStatus()`, `!OSLockStatus()` and `!SoftwareLockStatus()`, accesses to this register are **RW**.
- Otherwise, accesses to this register are **IMPDEF**.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xF00

PMITCTRL can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xF00

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register are **IMPDEF**.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

(old)

htmldiff from-

(new)

PMLAR, Performance Monitors Lock Access Register

The PMLAR characteristics are:

Purpose

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMLAR are RES0.

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses PMLAR to set or clear the lock, and PMU.PMLSR to check the current status of the lock. PMLSR to check the current status of the lock.

Attributes

PMLAR is a 32-bit register.

This register is part of the PMU block.

Field descriptions

When Software Lock is implemented:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																KEY															

KEY, bits [31:0]

Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface.

Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																RES0															

Otherwise

(old)

htmldiff from-

(new)

PMLSR, Performance Monitors Lock Status Register

The PMLSR characteristics are:

Purpose

Indicates the current status of the software lock for Performance Monitors registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented. Otherwise, direct accesses to PMLSR are RES0.

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses PMU.PMLAR to set or clear the lock, and PMLSR to check the current status of the lock. PMLAR to set or clear the lock, and PMLSR to check the current status of the lock.

Attributes

PMLSR is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																	nTT		SLK		SLI										

Bits [31:3]

Reserved, RES0.

nTT, bit [2]

Not thirty-two bit access required.

Reads as 0b0.

Access to this field is **RO**.

SLK, bit [1]

When Software Lock is implemented and FEAT_DoPD is not implemented:

Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when Software Lock is not implemented, this field is RES0.

For memory-mapped accesses when Software Lock is implemented, possible values of this field are:

SLK	Meaning
0b0	Lock clear. Writes are permitted to this component's registers.
0b1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

The reset behavior of this field is:

- On an External debug reset, this field resets to 1.

Otherwise:

Reserved, RAZ.

SLI, bit [0]

Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are:

SLI	Meaning
0b0	Software Lock not implemented or not memory-mapped access.
0b1	Software Lock implemented and memory-mapped access.

Accessing PMLSR

PMLSR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
PMU	0xFB4	PMLSR

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMLSR

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFB4

PMLSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xFB4

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMMIR, Performance Monitors Machine Identification Register

The PMMIR characteristics are:

Purpose

Describes Performance Monitors parameters specific to the implementation.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented and FEAT_PMUv3p4 is implemented. Otherwise, direct accesses to PMMIR are RES0.

PMMIR is in the Core power domain.

~~This register is present only when FEAT_PMUv3p4 is implemented. Otherwise, direct accesses to PMMIR are RES0.~~

Attributes

PMMIR is a ~~32-bit register.~~

- 64-bit register when FEAT_PMUv3_EXT64 is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32																
RES0																																															
RES0								EDGE								THWIDTH								BUS_WIDTH								BUS_SLOTS								SLOTS							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
RES0								THWIDTH								BUS_WIDTH								BUS_SLOTS								SLOTS															

Access to this field is **RO**.

THWIDTH, bits [23:20]

PMEVTYPER<n>_ELO PMU.PMEVTYPER<n>_TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the **PMEVTYPER<n>_ELO** feature, and, if implemented, the size of the PMU.PMEVTYPER<n>_TH field.

THWIDTH	Meaning
0b0000	FEAT_PMUv3_TH is not implemented.
0b0001	1 bit. PMU.PMEVTYPER<n>_TH[11:1] are PMEVTYPER<n>_ELO .TH[11:1] are RES0.
0b0010	2 bits. PMU.PMEVTYPER<n>_TH[11:2] are PMEVTYPER<n>_ELO .TH[11:2] are RES0.
0b0011	3 bits. PMU.PMEVTYPER<n>_TH[11:3] are PMEVTYPER<n>_ELO .TH[11:3] are RES0.
0b0100	4 bits. PMU.PMEVTYPER<n>_TH[11:4] are PMEVTYPER<n>_ELO .TH[11:4] are RES0.
0b0101	5 bits. PMU.PMEVTYPER<n>_TH[11:5] are PMEVTYPER<n>_ELO .TH[11:5] are RES0.
0b0110	6 bits. PMU.PMEVTYPER<n>_TH[11:6] are PMEVTYPER<n>_ELO .TH[11:6] are RES0.
0b0111	7 bits. PMU.PMEVTYPER<n>_TH[11:7] are PMEVTYPER<n>_ELO .TH[11:7] are RES0.
0b1000	8 bits. PMU.PMEVTYPER<n>_TH[11:8] are PMEVTYPER<n>_ELO .TH[11:8] are RES0.
0b1001	9 bits. PMU.PMEVTYPER<n>_TH[11:9] are PMEVTYPER<n>_ELO .TH[11:9] are RES0.
0b1010	10 bits. PMU.PMEVTYPER<n>_TH[11:10] are PMEVTYPER<n>_ELO .TH[11:10] are RES0.
0b1011	11 bits. PMU.PMEVTYPER<n>_TH[11] is PMEVTYPER<n>_ELO .TH[11] is RES0.
0b1100	12 bits.

All other values are reserved.

If FEAT_PMUv3_TH is not implemented, this field is zero.

Otherwise, the largest value that can be written to **PMU.PMEVTYPER<n>_TH** is $2^{\text{PMEVTYPER<n>_ELO.TH}}$ minus one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_WIDTH, bits [19:16]

Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\text{Log}_2(\text{number of bytes})$, plus one.

BUS_WIDTH	Meaning
0b0000	The information is not available.
0b0011	Four bytes.
0b0100	8 bytes.
0b0101	16 bytes.
0b0110	32 bytes.
0b0111	64 bytes.
0b1000	128 bytes.
0b1001	256 bytes.
0b1010	512 bytes.
0b1011	1024 bytes.
0b1100	2048 bytes.

All other values are reserved.

Each transfer is up to this number of bytes. An access might be smaller than the bus width.

When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.

When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS.

If the bus count information is not available, this field will read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SLOTS, bits [7:0]

Operation width. The largest value by which the STALL_SLOT event might increment **by** in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMMIR

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

PMMIR can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0				EDGE			THWIDTH				BUS_WIDTH				BUS_SLOTS				SLOTS												
Component										Offset										Instance											
PMU										0xE40										PMMIR											

This interface is accessible as follows:

- When !IsCorePowered(), or DoubleLockStatus(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

Bits [31:28]

Reserved, RES0.

EDGE, bits [27:24]

PMU event edge detection. Indicates implementation of the FEAT_PMUv3_EDGE feature.

EDGE	Meaning
0b0000	FEAT_PMUv3_EDGE is not implemented.
0b0001	FEAT_PMUv3_EDGE is implemented.

All other values are reserved.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

THWIDTH, bits [23:20]

PMU.PMEVTYPER<n>.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the PMU.PMEVTYPER<n>.TH field.

THWIDTH	Meaning
0b0000	FEAT_PMUv3_TH is not implemented.
0b0001	1 bit. PMU.PMEVTYPER<n>.TH[11:1] are RES0.
0b0010	2 bits. PMU.PMEVTYPER<n>.TH[11:2] are RES0.
0b0011	3 bits. PMU.PMEVTYPER<n>.TH[11:3] are RES0.
0b0100	4 bits. PMU.PMEVTYPER<n>.TH[11:4] are RES0.
0b0101	5 bits. PMU.PMEVTYPER<n>.TH[11:5] are RES0.
0b0110	6 bits. PMU.PMEVTYPER<n>.TH[11:6] are RES0.
0b0111	7 bits. PMU.PMEVTYPER<n>.TH[11:7] are RES0.
0b1000	8 bits. PMU.PMEVTYPER<n>.TH[11:8] are RES0.
0b1001	9 bits. PMU.PMEVTYPER<n>.TH[11:9] are RES0.
0b1010	10 bits. PMU.PMEVTYPER<n>.TH[11:10] are RES0.
0b1011	11 bits. PMU.PMEVTYPER<n>.TH[11] is RES0.
0b1100	12 bits.

All other values are reserved.

If FEAT_PMUv3_TH is not implemented, this field is zero.

Otherwise, the largest value that can be written to PMU.PMEVTYPER<n>.TH is $2^{(\text{PMMIR.THWIDTH})}$ minus one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_WIDTH, bits [19:16]

Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\text{Log}_2(\text{number of bytes})$, plus one.

BUS_WIDTH	Meaning
0b0000	The information is not available.
0b0011	Four bytes.
0b0100	8 bytes.
0b0101	16 bytes.
0b0110	32 bytes.
0b0111	64 bytes.
0b1000	128 bytes.
0b1001	256 bytes.
0b1010	512 bytes.
0b1011	1024 bytes.
0b1100	2048 bytes.

All other values are reserved.

Each transfer is up to this number of bytes. An access might be smaller than the bus width.

When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.

When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS.

If the bus count information is not available, this field will read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SLOTS, bits [7:0]

Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMMIR

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xE40

PMMIR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE40

- When !IsCorePowered(), or DoubleLockStatus(), or OSLockStatus() or !AllowExternalPMUAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RO**.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMOVS, Performance Monitors Overflow Flag Status register

The PMOVS characteristics are:

Purpose

The unsigned overflow flags for the Cycle Count Register, PMU.PMCCNTR_EL0, and each of the implemented event counters [PMEVCNTR<n>](#).

Configuration

External register PMOVS bits [31:0] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[31:0\]](#).

External register PMOVS bits [63:32] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMOVS bits [31:0] are architecturally mapped to AArch32 System register [PMOVSSET\[31:0\]](#).

External register PMOVS bits [31:0] are architecturally mapped to AArch64 System register [PMOVSCLR_EL0\[31:0\]](#).

External register PMOVS bits [63:32] are architecturally mapped to AArch64 System register [PMOVSCLR_EL0\[63:32\]](#) when FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented.

External register PMOVS bits [31:0] are architecturally mapped to AArch32 System register [PMOVS\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT64 is implemented. Otherwise, direct accesses to PMOVS are RES0.

Attributes

PMOVS is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															F0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, RES0.

F0, bit [32] When FEAT_PMUv3_ICNTR is implemented:

PMU.PMICNTR_EL0 unsigned overflow flag.

F0	Meaning
0b0	PMU.PMICNTR_EL0 has not overflowed.
0b1	PMU.PMICNTR_EL0 has overflowed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

C, bit [31]

Cycle counter unsigned overflow flag.

C	Meaning
0b0	The cycle counter has not overflowed since this bit was last cleared.
0b1	The cycle counter has overflowed since this bit was last cleared.

PMU.PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMU.PMCCNTR_EL0[31:0] or unsigned overflow of PMU.PMCCNTR_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter unsigned overflow bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	PMU.PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared.
0b1	PMU.PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared.

If FEAT_PMUv3p5 is implemented, [MDCR_EL2.HLP](#) and PMU.PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVS

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xC90

PMOVS can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC90

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus() and AllowExternalPMUAccess(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

no old file

htmldiff from-

(new)

The PMOVSLR_EL0 characteristics are:

Purpose

Contains the state of the overflow bit for the Cycle Count Register, `PMU.PMCCNTR_ELO`, and each of the implemented event counters, and each of the implemented event counters `PMEVCNTR<n>_ELO`, `PMEVCNTR<n>`, `PMCCNTR_ELO`. Writing to this register clears these bits.

Configuration

External register PMOVSLR_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMOVSLR_EL0\[31:0\]](#).

External register PMOVSLR_EL0 bits [63:32] are architecturally mapped to AArch64 System register PMOVSLR_EL0[63:32] when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented.

External register PMOVSLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMOVSRL\[31:0\]](#).

External register PMOVSLR_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[31:0\]](#).

External register PMOVSLR_EL0 bits [63:32] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[63:32\]](#) when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented.

External register PMOVSLR_EL0 bits [31:0] are architecturally mapped to AArch32 System register PMOVSSET[31:0].

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMOVSCLR_EL0 are RES0.

PMOVSLR EL0 is in the Core power domain.

Attributes

PMOVSLR_EL0 is a 32-bit register.

- 64-bit register when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT PMUv3 ICNTR is implemented:

6331623061296028592758265725562455235422532152205119501849174816471546144513441243114210419																							
RES0																							
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	

Bits [63:33]

Reserved, RES0.

F0, bit [32]**When FEAT_PMUv3_ICNTR is implemented:**

Unsigned overflow flag for PMU.PMICNTR_EL0 clear. On writes, allows software to clear the unsigned overflow flag for PMU.PMICNTR_EL0 to 0. On reads, returns the unsigned overflow flag for PMU.PMICNTR_EL0.

F0	Meaning
0b0	PMU.PMICNTR_EL0 has not overflowed.
0b1	PMU.PMICNTR_EL0 has overflowed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **W1C**.

Otherwise:

Reserved, RES0.

C, bit [31]

Cycle counter overflow clear bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.

PMCCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMCCNTR_EL0[31:0] or unsigned overflow of PMCCNTR_EL0. PMCCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMU.PMCCNTR_EL0[31:0] or unsigned overflow of PMU.PMCCNTR_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow clear bit for PMU.PMEVCNTR<n>_EL0. PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI. PMCFGR.N is less than 31, bits [30:PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect. PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, clears the PMU.PMEVCNTR<n>_EL0 overflow bit to 0. PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, clears the PMEVCNTR<n>_EL0 overflow bit to 0.

If FEAT_PMUv3p5 is implemented, MDCR_EL2.HLP and PMU.PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0]. PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMEVCNTR<n>_EL0[63:0].

- The reset behavior of this field is:
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSLR_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMOVSLR_EL0 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xC80								PMOVSLR_EL0															

- This interface is accessible as follows:
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
 - When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
 - Otherwise, accesses to this register generate an error response.

C, bit [31]

Cycle counter overflow clear bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.

PMU.PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMU.PMCCNTR_EL0[31:0] or unsigned overflow of PMU.PMCCNTR_EL0[63:0].

- The reset behavior of this field is:
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow clear bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, clears the PMU.PMEVCNTR<n>_EL0 overflow bit to 0.

If FEAT_PMUv3p5 is implemented, MDCR_EL2.HLP and PMU.PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSLR_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xC80

PMOVSLR_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xC80

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xC80

PMOVSLR_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC80	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xC84

PMOVSLR_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xC84	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

The PMOVSET_EL0 characteristics are:

Purpose

Sets the state of the overflow bit for the Cycle Count Register, `PMU.PMCCNTR_EL0`, and each of the implemented event counters `PMU.PMEVCNTR<n>_EL0`. `PMCCNTR_EL0`, and each of the implemented event counters `PMEVCNTR<n>_EL0`.

Configuration

External register PMOVSSET_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMOVSSET_EL0\[31:0\]](#).

External register PMOVSET_EL0 bits [63:32] are architecturally mapped to AArch64 System register [PMOVSET_EL0\[63:32\]](#) when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented.

External register PMOVSSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMOVSSET\[31:0\]](#).

External register PMOVSSSET_EL0 bits [31:0] are architecturally mapped to AArch64 System register PMOVSLR_EL0[31:0].

External register PMOVSET_EL0 bits [63:32] are architecturally mapped to AArch64 System register [PMOVSLR_EL0\[63:32\]](#) when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented.

External register PMOVSET_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMOVSr\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT is implemented. Otherwise, direct accesses to PMOVSSET_EL0 are RES0.

PMOVSSET_EL0 is in the Core power domain.

Attributes

PMOVSSET_EL0 is a 32-bit register.

- 64-bit register when FEAT_PMuV3_EXT64 is implemented, or FEAT_PMuV3p9 is implemented or FEAT_PMuV3_ICNTR is implemented
- 32-bit register otherwise

This register is part of the [PMU](#) block.

Field descriptions

When FEAT_PMUv3_EXT64 is implemented, or FEAT_PMUv3p9 is implemented or FEAT_PMUv3_ICNTR is implemented:

6331623061296028592758265725562455235422532152205119501849174816471546144513441243114210419																							
RES0																							
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	

Bits [63:33]

Reserved, RES0.

F0, bit [32]**When FEAT_PMUv3_ICNTR is implemented:**

Unsigned overflow flag for PMU.PMICNTR_EL0 set. On writes, allows software to set the unsigned overflow flag for PMU.PMICNTR_EL0 to 1. On reads, returns the unsigned overflow flag for PMU.PMICNTR_EL0.

F0	Meaning
0b0	PMU.PMICNTR_EL0 has not overflowed.
0b1	PMU.PMICNTR_EL0 has overflowed.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **W1S**.

Otherwise:

Reserved, RES0.

C, bit [31]

Cycle counter overflow set bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.

PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMCCNTR_EL0[31:0] or unsigned overflow of PMCCNTR_EL0. PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMU.PMCCNTR_EL0[31:0] or unsigned overflow of PMU.PMCCNTR_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow set bit for PMU.PMEVCNTR<n>_EL0. PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI. PMCFGR.N is less than 31, bits [30:PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect. PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, sets the PMU.PMEVCNTR<n>_EL0 overflow bit to 1. PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, sets the PMEVCNTR<n>_EL0 overflow bit to 1.

If FEAT_PMUv3p5 is implemented, MDCR_EL2.HLP and PMU.PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0]. PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMEVCNTR<n>_EL0[63:0].

- The reset behavior of this field is:
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSSET_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMOVSSET_EL0 can be accessed through the external debug interface:

Otherwise:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Component								Offset								Instance															
PMU								0xCC0								PMOVSSET_EL0															

- This interface is accessible as follows:
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
 - When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
 - Otherwise, accesses to this register generate an error response.

C, bit [31]

Cycle counter overflow set bit.

C	Meaning
0b0	When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.

PMU.PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of PMU.PMCCNTR_EL0[31:0] or unsigned overflow of PMU.PMCCNTR_EL0[63:0].

- The reset behavior of this field is:
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

P<n>, bit [n], for n = 30 to 0

Event counter overflow set bit for PMU.PMEVCNTR<n>_EL0.

If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are RAZ/WI.

P<n>	Meaning
0b0	When read, means that PMU.PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect.
0b1	When read, means that PMU.PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, sets the PMU.PMEVCNTR<n>_EL0 overflow bit to 1.

If FEAT_PMUv3p5 is implemented, MDCR_EL2.HLP and PMU.PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of PMU.PMEVCNTR<n>_EL0[31:0] or unsigned overflow of PMU.PMEVCNTR<n>_EL0[63:0].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

Accessing PMOVSSET_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

When FEAT_PMUv3_EXT64 is implemented

BlockAccess at address 0xCC0

PMOVSSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xCC0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When FEAT_PMUv3_EXT32 is implemented

BlockAccess at address 0xCC0

PMOVSSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xCC0	31:0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

When (FEAT_PMUv3_EXT32 is implemented and FEAT_PMUv3p9 is implemented) or FEAT_PMUv3_ICNTR is implemented

BlockAccess at address 0xCC4

PMOVSSET_EL0 can be accessed through the PMU block as follows:

Frame	Offset	Range
PMU	0xCC4	63:32

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **RW**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

no old file

htmldiff from-

(new)

PMPCSCCTL, PC Sample-based Profiling Control Register

The PMPCSCCTL characteristics are:

Purpose

Controls the PC Sample-based Profiling feature.

Configuration

This register is present only when FEAT_PCSRv8p9 is implemented. Otherwise, direct accesses to PMPCSCCTL are RES0.

PMPCSCCTL is in the Core power domain.

Attributes

PMPCSCCTL is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																															
RES0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:5]

Reserved, RES0.

SS, bit [4]

When FEAT_PMUv3_SS is implemented:

Sample on Snapshot.

Controls whether the following registers are sampled on a PMU snapshot Capture event:

PMU.PMCID1SR, PMU.PMCID2SR, and PMU.PMPCSR.

SS	Meaning
0b0	Sample on Read.
0b1	Sample on Snapshot.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [3:2]

Reserved, RES0.

IMP, bit [1]

Profiling enable implemented.

IMP	Meaning
0b0	PMPCSCTL.EN reads-as-zero and ignores writes.
0b1	PMPCSCTL.EN is a read-write control bit.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

EN, bit [0]**When PMU.PMPCSCTL.IMP == 1:**

PC Sample-based Profiling Enable.

EN	Meaning
0b0	PC Sample-based Profiling is suspended.
0b1	PC Sample-based Profiling is active.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RAZ/WI.

Accessing PMPCSCTL

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x230

PMPCSCTL can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x230

- When DoubleLockStatus(), or !IsCorePowered() or OSLockStatus(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMPCSR, Program Counter Sample Register

The PMPCSR characteristics are:

Purpose

Holds a sampled instruction address value.

Configuration

This register is present only when FEAT_PMUv3_EXT is implemented and FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMPCSR are RES0.

PMPCSR is in the Core power domain.

~~This register is present only when FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMPCSR are RES0.~~

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID.PCSample](#).

Support for 64-bit atomic reads is IMPLEMENTATION DEFINED. If 64-bit atomic reads are implemented, a 64-bit read of PMPCSR has the same side-effect as a 32-bit read of PMCSR[31:0] followed by a 32-bit read of PMPCSR[63:32], returning the combined value. For example, if the PE is in Debug state then a 64-bit atomic read returns bits[31:0] == 0xFFFFFFFF and bits[63:32] UNKNOWN.

Attributes

PMPCSR is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
NS	EL	T	NSE	RES0	PCSample[55:32]																											
PCSample[31:0]																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

NS, bit [63]

When FEAT_RME is implemented:

Together with the NSE field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Otherwise:

Non-secure state sample. Indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

If EL3 is not implemented, this bit indicates the Effective value of SCR.NS.

NS	Meaning
0b0	Sample is from Secure state.
0b1	Sample is from Non-secure state.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

EL, bits [62:61]

Exception level status sample. Indicates the Exception level that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

EL	Meaning
0b00	Sample is from EL0.
0b01	Sample is from EL1.
0b10	Sample is from EL2.
0b11	Sample is from EL3.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

T, bit [60]**When FEAT_TME is implemented:**

Transactional state of the sample. Indicates the Transactional state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

T	Meaning
0b0	Sample is from Non-transactional state.
0b1	Sample is from Transactional state.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

NSE, bit [59]**When FEAT_RME is implemented:**

Together with the NS field, indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample.

For a description of the values derived by evaluating NS and NSE together, see PMPCSR.NS.

Otherwise:

Reserved, RES0.

Bits [58:56]

Reserved, RES0.

PCSample[55:32], bits [55:32]

Bits[55:32] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

PCSample[31:0], bits [31:0]

Bits[31:0] of the sampled instruction address value.

PMPCSR[31:0] reads as 0xFFFFFFFF when any of the following are true:

- The PE is in Debug state.
- PC Sample-based profiling is prohibited.

If a branch instruction has retired since the PE left reset state, then the first read of PMPCSR[31:0] is permitted but not required to return 0xFFFFFFFF.

PMPCSR[31:0] reads as an UNKNOWN value when any of the following are true:

- The PE is in reset state.
- No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited.
- No branch instruction has retired since the last read of PMPCSR[31:0].

For the cases where a read of PMPCSR[31:0] returns 0xFFFFFFFF or an UNKNOWN value, the read has the side-effect of setting PMPCSR[63:32], PMU.PMCID1SR, PMU.PMCID2SR, and PMU.PMVIDSR to ~~PMCID1SR, PMCID2SR, and PMVIDSR~~ to UNKNOWN values.

Otherwise, a read of PMPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to PMPCSR[63:32], PMU.PMCID1SR, PMU.PMCID2SR, and PMU.PMVIDSR. ~~The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}. PMCID1SR, PMCID2SR, and PMVIDSR. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.~~

For a read of PMPCSR[31:0] from the memory-mapped interface, if PMLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the side-effect of the access does not occur and PMPCSR[63:32], PMU.PMCID1SR, PMU.PMCID2SR, and PMU.PMVIDSR are unchanged. ~~PMCID1SR, PMCID2SR, and PMVIDSR are unchanged.~~

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing PMPCSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Note

~~A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.~~

PMPCSR can be accessed through the external debug interface:

Component	Offset	Instance	Range
-----------	--------	----------	-------

PMU	0x200	PMPCSR	31:0
-----	-------	--------	------

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
PMU	0x204	PMPCSR	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
PMU	0x220	PMPCSR	31:0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Component	Offset	Instance	Range
PMU	0x224	PMPCSR	63:32

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMPCSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Note

A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x200

PMPCSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x200

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

BlockAccess at address 0x220

PMPCSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x220

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5707; 21c5a6dd0fdaf10a712e2f2d6ffbbdd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)
-------	----------------	-------

PMPIDR0, Performance Monitors Peripheral Identification Register 0

The PMPIDR0 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configuration

This Implementation register of isthis presentregister only when FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR0. Otherwise, direct accesses to PMPIDR0 are RES0OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMPIDR0 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								PART_0							

Bits [31:8]

Reserved, RES0.

PART_0, bits [7:0]

Part number, least significant byte.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMPIDR0

PMPIDR0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFE0	PMPIDR0

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMPIDR0

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFE0

PMPIDR0 can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xFE0

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

3005/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMPIDR1, Performance Monitors Peripheral Identification Register 1

The PMPIDR1 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configuration

ThisImplementation registerof isthis presentregister only when FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR1. Otherwise, direct accesses to PMPIDR1 are RES0OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMPIDR1 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								DES_0				PART_1			

Bits [31:8]

Reserved, RES0.

DES_0, bits [7:4]

Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 0b1011.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PART_1, bits [3:0]

Part number, most significant nibble.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

PMPIDR2, Performance Monitors Peripheral Identification Register 2

The PMPIDR2 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configuration

ThisImplementation registerof isthis presentregister only when FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR2. Otherwise, direct accesses to PMPIDR2 are RES0OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMPIDR2 is a 32-bit register.

This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								REVISION			JEDEC		DES_1		

Bits [31:8]

Reserved, RES0.

REVISION, bits [7:4]

Part major revision. Parts can also use this field to extend Part number to 16-bits.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

JEDEC, bit [3]

Indicates a JEP106 identity code is used.

Reads as 0b1.

Access to this field is **RO**.

Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Accessing PMPIDR2

PMPIDR2 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xFE8	PMPIDR2

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMPIDR2

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xFE8

PMPIDR2 can be accessed through the PMU block as follows:

	Frame		Offset
	PMU		0xFE8

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

PMPIDR3, Performance Monitors Peripheral Identification Register 3

The PMPIDR3 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.
For more information, see 'About the Peripheral identification scheme'.

Configuration

ThisImplementation registerof isthis presentregister only when FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR3. Otherwise, direct accesses to PMPIDR3 are RES0OPTIONAL.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.
This register is required for CoreSight compliance.

Attributes

PMPIDR3 is a 32-bit register.
This register is part of the PMU block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																								REVAND				CMOD			

Bits [31:8]
Reserved, RES0.

REVAND, bits [7:4]
Part minor revision. Parts using PMU.PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.~~PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.~~
This field has an IMPLEMENTATION DEFINED value.
Access to this field is **RO**.

CMOD, bits [3:0]
Customer modified. Indicates someone other than the Designer has modified the component.
This field has an IMPLEMENTATION DEFINED value.
Access to this field is **RO**.

(old)

htmldiff from-

(new)

PMPIDR4, Performance Monitors Peripheral Identification Register 4

The PMPIDR4 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configuration

This **Implementation** register of **this** present register only when FEAT_PMUv3_EXT is implemented and an implementation implements PMPIDR4. Otherwise, direct accesses to PMPIDR4 are **RES0** **OPTIONAL**.

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

PMPIDR4 is a 32-bit register.

This register is part of the **PMU** block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
												RES0																SIZE								DES_2			

Bits [31:8]

Reserved, RES0.

SIZE, bits [7:4]

Size of the component. Log₂ of the number of 4KB pages from the start of the component to the end of the component ID registers.

Reads as 0b0000.

Access to this field is **RO**.

DES_2, bits [3:0]

Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

no old file

htmldiff from-

(new)

PMSSCR_EL1, Performance Monitors Snapshot Status and Capture Register

The PMSSCR_EL1 characteristics are:

Purpose

Holds status information about the captured counters and provides a mechanism for software to initiate a sample.

Configuration

This register is present only when FEAT_PMUv3_SS is implemented. Otherwise, direct accesses to PMSSCR_EL1 are RES0.

Attributes

PMSSCR_EL1 is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
																RES0																NC
																RES0																SS
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Bits [63:33]

Reserved, RES0.

NC, bit [32]

No Capture. Indicates whether the PMU counters have been captured.

NC	Meaning
0b0	PMU counters captured.
0b1	PMU counters not captured.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Bits [31:1]

Reserved, RES0.

SS, bit [0]

Snapshot Capture and Status.

SS	Meaning
0b0	On a read: The Capture event has completed. On a write: Ignored.
0b1	On a read: The Capture event has not completed. On a write: Initiate a capture immediately.

It is CONSTRAINED UNPREDICTABLE whether a Capture event has completed if this field is modified when the Capture event is ongoing.

Note

If FEAT_Debugv8p4 is implemented, the OPTIONAL Software Lock is not implemented.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing this field has the following behavior:

- When SoftwareLockStatus(), access to this field is **RO**.
- Otherwise, access to this field is **WO/RAZ**.

Accessing PMSSCR_EL1

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xE30

PMSSCR_EL1 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xE30

- When !AllowExternalPMSSAccess(), accesses to this register generate an error response.
- Otherwise, accesses to this register are **RW**.

30/09/2022 15:57; 21c5a6dd0fdaf10a712e2f2d6fffbdbd66d4d96f

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSWINC_EL0, Performance Monitors Software Increment register

The PMSWINC_EL0 characteristics are:

Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW_INCR'.

Configuration

External register PMSWINC_EL0 bits [31:0] are architecturally mapped to AArch64 System register [PMSWINC_EL0\[31:0\]](#).

External register PMSWINC_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMSWINC\[31:0\]](#).

This register is present only when FEAT_PMUv3_EXT32 is implemented and an implementation implements PMSWINC_EL0. Otherwise, direct accesses to PMSWINC_EL0 are RES0.

PMSWINC_EL0 is in the Core power domain.

Implementation of this register is OPTIONAL.

If this register is implemented, use of it is deprecated.

If 1 is written to bit [n] from the external debug interface, it is CONSTRAINED UNPREDICTABLE whether or not a SW_INCR event is created for counter n. This is consistent with not implementing the register in the external debug interface.

Attributes

PMSWINC_EL0 is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Bit [31]

Reserved, RES0.

P<n>, bit [n], for n = 30 to 0

Event counter software increment bit for [PMU.PMEVCNTR<n>_EL0](#). [PMEVCNTR<n>_EL0](#).

If [PMU.PMCFGR.N](#) is less than 31, bits [30:PMU.PMCFGR.N] are WI. [PMCFGR.N](#) is less than 31, bits [30:PMCFGR.N] are WI.

P<n>	Meaning
0b0	No action. The write to this bit is ignored.
0b1	It is CONSTRAINED UNPREDICTABLE whether a SW_INCR event is generated for event counter n.

Accessing PMSWINC_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMSWINC_EL0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xCA0	PMSWINC_EL0

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **WI**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **WO**.
- Otherwise, accesses to this register generate an error response.

Accessing PMSWINC_EL0

Note

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0xCA0

PMSWINC_EL0 can be accessed through the PMU block as follows:

Frame	Offset
PMU	0xCA0

- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and SoftwareLockStatus(), accesses to this register are **WI**.
- When IsCorePowered(), !DoubleLockStatus(), !OSLockStatus(), AllowExternalPMUAccess() and !SoftwareLockStatus(), accesses to this register are **WO**.
- Otherwise, accesses to this register generate an error response.

3095/0907/2022 1517:5708; 21c5a6dd0fdaf10a712e2f2d6ffbd6d66d4d96fb0421fa9a8865165f9b91af9b4a566111f866305

Copyright © 2010-2022 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

PMVCIDSR, CONTEXTIDR_EL1 and VMID Sample Register

The PMVCIDSR characteristics are:

Purpose

Contains the sampled CONTEXTIDR_EL1 and VMID values that are captured on reading PMU.PMPCSR.

Configuration

This register is present only when FEAT_PMUv3_EXT64 is implemented and FEAT_PCSRv8p2 is implemented. Otherwise, direct accesses to PMVCIDSR are RES0.

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID](#).PCSample.

Attributes

PMVCIDSR is a 64-bit register.

This register is part of the [PMU](#) block.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
RES0																VMID[15:8]								VMID							
CONTEXTIDR_EL1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

VMID[15:8], bits [47:40]

When FEAT_VMID16 is implemented:

Extension to VMID[7:0]. For more information, see VMID[7:0].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VMID, bits [39:32]

VMID sample. The VMID associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample was generated:

- This field is set to an UNKNOWN value if any of the following apply:
 - EL2 is disabled in the current Security state.
 - The PE is executing at EL2.
 - EL2 is enabled in the current Security state, the PE is executing at EL0, EL2 is using AArch64, HCR_EL2.E2H == 1, and HCR_EL2.TGE == 1.
- Otherwise:
 - If EL2 is using AArch64 and either FEAT_VMID16 is not implemented or [VTCR_EL2.VS](#) is 1, this field is set to [VTTBR_EL2.VMID](#).
 - If EL2 is using AArch64, FEAT_VMID16 is implemented, and [VTCR_EL2.VS](#) is 0, PMVIDSR.VMID[7:0] is set to [VTTBR_EL2.VMID\[7:0\]](#) and PMVIDSR.VMID[15:8] is RES0.
 - If EL2 is using AArch32, this field is set to [VTTBR.VMID](#).

Because the value written to PMVIDSR is an indirect read of the VMID value, it is CONSTRAINED UNPREDICTABLE whether PMVIDSR is set to the original or new value if PMU.PMPCSR samples:

- An instruction that writes to the VMID value.
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

CONTEXTIDR_EL1, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample is generated:

- If EL1 is using AArch64, then the Context ID is sampled from [CONTEXTIDR_EL1](#).
- If EL1 is using AArch32, then the Context ID is sampled from [CONTEXTIDR](#).
- If EL3 is implemented and is using AArch32, then [CONTEXTIDR](#) is a banked register and this register samples the current banked copy of [CONTEXTIDR](#) for the Security state that is associated with the most recent PMU.PMPCSR sample.

Because the value written to this register is an indirect read of CONTEXTIDR, it is CONSTRAINED UNPREDICTABLE whether this register is set to the original or new value if PMU.PMPCSR samples:

- An instruction that writes to CONTEXTIDR.
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing PMVCIDSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x208

PMVCIDSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x208

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are **RO**.
- Otherwise, accesses to this register generate an error response.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMVIDSR, VMID Sample Register

The PMVIDSR characteristics are:

Purpose

Contains the sampled VMID value that is captured on reading [PMU.PMPCSR\[31:0\]](#). [PMPCSR\[31:0\]](#).

Configuration

This register is present only when FEAT_PMUv3_EXT32 is implemented, FEAT_PCSRv8p2 is implemented and EL2 is implemented. Otherwise, direct accesses to PMVIDSR are RES0.

PMVIDSR is in the Core power domain.

This register is present only when FEAT_PCSRv8p2 is implemented and EL2 is implemented. Otherwise, direct accesses to PMVIDSR are RES0.

Note

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of [EDDEVID.PCSample](#).

Attributes

PMVIDSR is a 32-bit register.

This register is part of the [PMU](#) block.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																VMID[15:8]								VMID							

Bits [31:16]

Reserved, RES0.

VMID[15:8], bits [15:8]

When FEAT_VMID16 is implemented:

Extension to VMID[7:0]. For more information, see VMID[7:0].

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

VMID, bits [7:0]

VMID sample. The VMID associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample was generated: PMPCSR sample. When the most recent PMPCSR sample was generated:

- This field is set to an UNKNOWN value if any of the following apply:
 - EL2 is disabled in the current Security state.
 - The PE is executing at EL2.
 - EL2 is enabled in the current Security state, the PE is executing at EL0, EL2 is using AArch64, HCR_EL2.E2H == 1, and HCR_EL2.TGE == 1.
- Otherwise:
 - If EL2 is using AArch64 and either FEAT_VMID16 is not implemented or VTCR_EL2.VS is 1, this field is set to VTTBR_EL2.VMID.
 - If EL2 is using AArch64, FEAT_VMID16 is implemented, and VTCR_EL2.VS is 0, PMVIDSR.VMID[7:0] is set to VTTBR_EL2.VMID[7:0] and PMVIDSR.VMID[15:8] is RES0.
 - If EL2 is using AArch32, this field is set to VTTBR.VMID.

Because the value written to PMVIDSR is an indirect read of the VMID value, it is CONSTRAINED UNPREDICTABLE whether PMVIDSR is set to the original or new value if PMU.PMPCSR samples: PMPCSR samples:

- An instruction that writes to the VMID value.
- The next Context synchronization event.
- Any instruction executed between these two instructions.

The reset behavior of this field is:

- On a Cold reset, this field resets to an architecturally UNKNOWN value.

Accessing PMVIDSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

PMVIDSR can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x20C	PMVIDSR

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are RO.
- Otherwise, accesses to this register generate an error response.

Accessing PMVIDSR

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Accesses to this register use the following encodings in the external debug interface:

BlockAccess at address 0x20C

PMVIDSR can be accessed through the PMU block as follows:

Frame	Offset
PMU	0x20C

- When IsCorePowered(), !DoubleLockStatus() and !OSLockStatus(), accesses to this register are RO.
- Otherwise, accesses to this register generate an error response.

(old)

htmldiff from-

(new)